TUGAS AKHIR – TE 141599

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Muhammad Soleh Gangsarestu
NRP 2211100156

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Ronny Mardiyanto, ST., MT., Ph.D.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

TUGAS AKHIR – TE 141599

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Muhammad Soleh Gangsarestu
NRP 2211100156

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Ronny Mardiyanto, ST., MT., Ph.D.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

DESIGN AND REALIZATION OF MEN'S CLOTHES SIZE MEASURING SYSTEM BASED ON KINECT

Muhammad Soleh Gangsarestu
NRP 2211100156

Supervisor
Dr. Ir. Djoko Purwanto, M.Eng.
Ronny Mardiyanto, ST., MT., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institute Technology of Sepuluh Nopember
Surabaya 2015

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Ir. Djoko Purwanto, M.Eng.	Ronny Mardiyanto, ST., MT., Ph.D.
NIP.195004021986012001	NIP. 198101182003121003

SURABAYA
JULI, 2015

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Muhammad Soleh Gangsarestu
2211100156

Dosen Pembimbing 1 : Dr. Ir. Djoko Purwanto, M.Eng
Dosen Pembimbing 2 : Ronny Mardiyanto, ST., MT.,Ph.D

ABSTRAK

Computer Vision, bidang yang melibatkan metode proses, akuisisi, analisa, dan pengolahan citra dari dunia nyata untuk menghasilkan informasi numerik maupun simbol. Informasi itu dapat digunakan untuk melakukan suatu interpretasi satuan panjang. Interpretasi tersebut dapat dimanfaatkan pada toko baju untuk sebuah sistem pengukuran badan seseorang. Sistem ini dikhususkan untuk mengukur badan pria karena lebih mudah daripada mengukur badan wanita. Penggunaannya dapat mengurangi kerugian baju yang longgar akibat dicoba oleh pelanggan, serta pelanggan merasa puas dan nyaman karena proses pengukuran yang cepat dan mendapatkan ukuran yang sesuai.

Pada Tugas Akhir dilakukan rancang bangun sistem pengukuran badan pria untuk menentukan ukuran baju. Sistem ini akan dilakukan pada suatu ruangan yang dilengkapi dengan sebuah kamera kinect, layar monitor, dan unit proses untuk interaksi pengguna dengan sistem. Kinect camera digunakan karena dapat mengambil citra RGB dan dilengkapi dengan IR *projector* serta IR *sensor* yang dapat memberikan informasi *depth*. Setelah pengguna memulai sistem, kamera kinect akan mengirimkan informasi gambar serta *depth* ke unit proses untuk melakukan pengolahan informasi visual. *Neural networks* digunakan untuk estimasi lebar badan dari pengguna, selanjutnya sistem akan memberikan saran ukuran baju S, M, L, XL, dan - (tidak ada ukuran baju yang sesuai). Pada layar monitor akan memperlihatkan data hasil pengukuran serta kategori ukuran baju yang dianjurkan untuk pengguna. Berdasarkan pengujian yang dilakukan estimasi lebar badan memilki nilai error absolut maksimal sebesar 7,95% pada jarak optimal (1500 mm).

Kata Kunci : Baju, *Computer Vision,* Kamera Kinect

DESIGN AND REALIZATION OF MEN'S CLOTHES SIZE MEASURING SYSTEM BASED ON KINECT

Muhammad Soleh Gangsarestu
2211100156

Supervisor : Dr. Ir. Djoko Purwanto, M.Eng
Co-Supervisor : Ronny Mardiyanto, ST.,MT.,Ph.D

ABSTRACT

Computer Vision is a field of study which containing some method such as acquisition, analyzing and processing of image. From those method it gives numerical and symbolic information. Those information can be used for interpret length unit. It can be used in clothing store for measuring customer's body. This system is specified for measuring men's body because it is easier than measuring women's body. This system can reduce the clothing store loss by reducing the ammount of clothes being loosen up because of the customers trials and customer will be satisfied because of the fast measuring process and have the appropriate size to the customer's body.

The Final Project designs a system which measure mens's body to determine clothes size. This body measurement system will be done in a room that is equpipped with kinect camera, monitor display, and a processing unit for the human machine interaction. Kinect camera is used because it is not only cappable of taking RGB image but also equipped with IR projector and sensor that gives depth information. After the user start the measurement system, the kinect camera will send the visual information with depth to processing unit to perform the processing of visual ingormation. Neural networks are used to estimate user's body width and then sytem will suggest shirt size category S, M, L, XL and - (there is no suitable shirt) for the user. At the monitor display will show the measurement data result and at what size category will the shirt be suggested for the user. Based on test that have been conducted the estimate chest width have maximum absolute error of 7,95% within system's optimal range (1500mm).

***Keywords**: Computer Vision, Kinect Camera, Shirt*

KATA PENGANTAR

Alhamdulillah, Puji syukur kepada Allah SWT atas kasih dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Dalam mengerjakan Tugas Akhir ini tentunya penulis mendapatkan banyak bantuan berupa saran, dorongan semangat, dan doa. Atas semua hal itu penulis mengucapkan banyak terimakasih dan semoga Allah akan membalas dengan sebaik-baiknya balasan. Dan sekali lagi saya ucapkan terima kasih kepada:

1. Bapak, Ibu, kakak-kakak serta seluruh keluarga yang telah memberikan dukungan semangat, motivasi, saran dan doa.
2. Bapak Dr. Ir. Djoko Purwanto, M. Eng., selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, yang diberikan selama pengerjan penelitian tugas akhir ini.
3. Bapak Ronny Mardiyanto, ST., MT., Ph. D., selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan,
4. Mas Anhar, yang telah membantu memberikan ide, serta saran-saran dalam pengerjaan Tugas Akhir ini
5. Mas Eja, yang telah meminjamkan kamera Kincet dan mengajari cara awal penggunaan kamera kinect selama pengerjaan Tugas Akhir ini
6. Mas Toni, yang turut memberikan saran dalam pengerjaan Tugas Akhir ini.
7. Mas Hilton, yang memberikan pengarahan pada awal pengerjaan Tugas Akhir.
8. Reza yang telah membantu saya diawal pengerjaan Tugas Akhir ini
9. Serta rekan-rekan seluruh asisten laboratorium elektronika yang telah membantu saya untuk pengambilan data dan pengujian data.

Surabaya, Juli 2015

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PUSTAKA

[1] N. Dao, T. Deng, and J. Cai, “Fast and automatic body circular measurement based on a single kinect,” in Asia-Pacific Signal and Information Processing Association, IEEE Annual Summit and Conference (APSIPA), 2014, pp.1-4.

[2] P. Roy, J. Nancy Staples, and J. Steve Davis, “Automatic measurement extraction for apparel from a three-dimensional body scan," in Optics and Lasers in Engineering, Elsevier Science Limited Vol. 28, 1997, pp.157-172.

[3] B. Lv, and Q. Sun Shou “Automatic measurement of scanned human body in fixed posture” in Computer-Aided Industrial Design & Conceptual Design (CAIDCD), IEEE 11th International Conference Vol. 1, 2010, pp. 575-578.

[4] Dorland, W. A. Newman, “Dorland’s Illustrated Medical Dictionary 32nd Edition,” Elsevier Health Sciences, 2011 pp.99

[5] National Aeronautics and Space Administration (NASA), “Anthropometry and Biomechanics”. <URL: http://msis.jsc.nasa.gov/sections/section03.htm> Maret 2015.

[6] Microsoft Developer Network Library. “Kinect for Windows Sensor Components and Specifications”. <URL: http://msdn.microsoft.com/en-us/library/jj131033.aspx>Maret, 2015.

[7] Z. Zhengyou, “Microsoft Kinect Sensor and Its Effect,” MultiMedia, IEEE, Vol. 19 No.2 2012, pp.4-10.

[8] H. Jungong, S. Ling, X. Dong, and J. Shotton, "Enhanced Computer Vision with Microsoft Kinect Sensor,” in Cybernetics, IEEE Transactions Vol. 43, No. 5 2013, pp. 1318-1334.

[9] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, “Depth Mapping Using Projected Patterns,” in U.S. Patent, No 8, 2008.

[10] K. Khoshelham, “Accuracy Analysis of Kinect Depth Data,” in ISPRS Workshop Laser Scanning, Vol. 38 No. 5, 2011, pp. W12.

[11] J.Shotton, A. Fitzgibbon, M. Cook, and T. Sharp, “Real-Time Human Pose Recognition in Parts from Single Depth Images,” in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference, 2011, pp. 1297-1304.

[12] S. Haykin, “Neural Networks A Comprehensive Foundation 2nd Edition” Prentice Hall, 1994 pp. 24.

[13] M. Frederic Ham, I. Konstanic, “Principles Of Neurocomputing for science & engineering,” McGraw-Hill Higher Education, 2000. pp 26-110.
[14] M. M. Deza and E. Deza “Encyclopedia of Distances 3rd Edition,” Springer Berlin Heidelberg, 2014, pp. 100
[15] Visual Studio Developer. “Visual Studio - Microsoft Developer Tools”. <URL: https://www.visualstudio.com> Mei, 2015.
[16] OpenCV Developers Team. “About”. <URL: http://opencv.org/about.html > Mei, 2015.
[17] MathWorks Technical Supports. “MATLAB” <URL: http://www.mathworks.com/products/matlab/features-b.html> Mei 2015
[18] MathWorks Technical Supports. “Improve Neural Network Generalization and Avoid Overfitting” <URL: http://www.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html> Mei 2015

BIODATA PENULIS

Muhammad Soleh Gangsarestu, seseorang yang akrab dipanggil Restu, lahir di Surabaya pada 21 Februari 1993. Penulis merupakan anak keempat dari empat bersaudara. Selama ini penulis telah menempuh pendidikan di SD Negeri 02 Pulo Gebang Jakarta (1999-2005) SMP Negeri 172 Jakarta (2005-2008) SMA Negeri 103 Jakarta (2008-2011) dan terakhir sebagai mahasiswa S1 Teknik Elektro khususnya bidang studi elektronika di ITS Surabaya (2011-2015).

Selama masa kuliah penulis aktif berorganisasi dalam UKM Fotografi ITS sebagai wakil ketua periode 2013-2014, dan menjadi koordinator Laboratorium Elektronika Dasar B 202 periode 2014-2015.

Email:
m.soleh.gangsarestu@gmail.com

BAB I
PENDAHULUAN

1.1 Latar Belakang

Setiap toko pakaian menjual berbagai macam jenis pakaian yang terbuat dari berbagai jenis bahan. Bahan yang digunakan tergantung dari penggunaan pakaian tersebut. Bahan-bahan tersebut diantaranya katun wol, sutra, linen, spandex dan poliester. Untuk melayani pelanggan maka, toko tersebut menyediakan ruang ganti untuk mencoba pakaian yang ada. Tetapi tidak semua pakaian yang dijual dapat dicoba secara langsung oleh pelanggan. Terutama kaus yang menggunakan bahan katun.

Adapun alasan yang menyebabkan kaus berbahan katun tidak diperbolehkan untuk dicoba secara langsung karena mudah menjadi longgar setelah digunakan oleh beberapa orang yang berbeda. Hal ini tentu saja menimbulkan masalah di kedua belah pihak, yaitu pihak penjual seperti toko baju dan pihak pengunjung toko. Bagi pemilik toko baju, baju yang longgar tersebut tidak layak untuk dijual, bahkan untuk dipajang juga dirasa kurang pantas. Bila hal ini terus berlanjut tentunya pihak pemilik toko akan mengalami kerugian yang cukup besar akibat dari baju yang tidak dapat dijual tersebut. Sedangkan disisi pelanggan, mereka akan merasa khawatir bila tidak bisa mencoba pakaian tersebut secara langsung, dan setelah mereka membeli dan mencoba pakaian tersebut ternyata ukuran yang mereka pilih tidak sesuai. Selain itu bila pihak toko menawarkan untuk mengukur ukuran badan pelanggan, maka pengukurannya juga masih menggunakan pita meter baju. Tidak semua pelanggan merasa nyaman dengan cara pengukuran tersebut[1]. Untuk memberikan kenyamanan yang menguntungkan kedua belah pihak baik penjual maupun pembeli, maka perlu adanya informasi tambahan mengenai ukuran dari baju tersebut[2][3].

Atas dasar kesulitan penentuan ukuran badan calon pembeli, dalam Tugas Akhir akan dilakukan rancang bangun sistem pengukuran badan pria untuk menentukan ukuran baju. Sasaran dari sistem pengukuran ini yaitu calon pembeli pria yang tidak mengetahui ukuran badannya. Mereka dapat masuk ke suatu ruangan yang dilengkapi dengan sebuah kamera kinect dan sebuah layar monitor. Untuk mendapatkan ukuran badan, pengguna berdiri didepan kamera kinect pada jarak 1.5 meter. Selanjutnya

informasi data citra akan diambil untuk keperluan pengolahan citra oleh sebuah unit proses. Dengan memanfaatkan informasi *depth* yang diambil oleh kamera kinect maka lebar badan pengguna dapat diperkirakan. Dan pada akhirnya hasil pengukuran serta kategori ukuran baju S, M, L, XL dan - (tidak ada ukuran baju yang sesuai) akan ditampilkan pada layar monitor.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimana melakukan estimasi lebar badan menggunakan informasi visual yang didapat dari kamera Kinect.
2. Bagaimana cara memberikan kategori ukuran baju berdasarkan estimasi pengukuran yang didapat.
3. Bagaimana mengkomunikasikan hasil pengukuran yang diperoleh dari pemrosesan citra kepada pengguna dengan baik.

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Mampu melakukan estimasi lebar badan menggunakan kamera Kinect.
2. Mampu memberikan kategori ukuran baju berdasarkan estimasi pengukuran yang didapat.
3. Mampu mengkomunikasikan hasil pengukuran yang diperoleh dari pemrosesan citra kepada pengguna dengan baik.

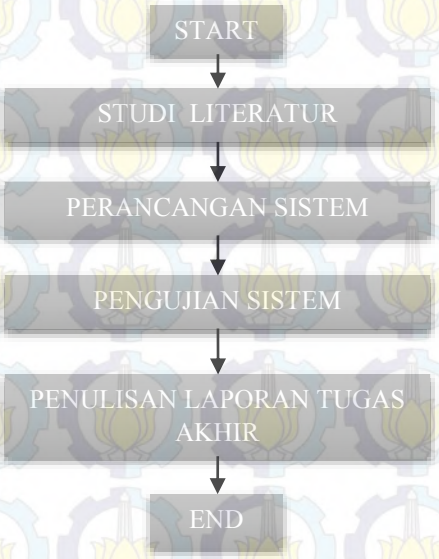
1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Jenis baju yang direkomendasikan yaitu berjenis kaus atau kaus berkerah.
2. Tinggi kamera Kinect terhadap lantai ± 1 meter
3. Jarak yang dibutuhkan untuk pengukuran ±1.5 meter.
4. Posisi badan tegak lurus dengan kamera.
5. Posisi bahu lurus dan sejajar.
6. Pengkuran dilakukan pada posisi yang diperlihatkan oleh program *interface.*
7. Menggunakan kaus, kaus berkerah, atau kemeja saat pengukuran.
8. Tidak bergerak selama pengukuran dilakukan.
9. Tidak ada orang lain pada area pengukuran.

1.5 Metodologi Penelitian

Dalam penyelesaian Tugas Akhir ini digunakan metodologi sebagai berikut :



Gambar 1.1 Metodologi Tugas Akhir

1. **Studi Literatur**

   Pada tahap studi literatur dilakukan pengumpulan teori dan data untuk digunakan pada Tugas Akhir. Teori dan data yang digunakan diambil dari buku-buku, jurnal, proceeding, dan artikel-artikel di internet, meliputi:

   - Konfigurasi kamera kinect dengan Visual Studio 2010.
   - Penggunaan *library* OpenCV pada pemrosesan citra kamera kinect.
   - Mempelajari bagaimana kamera kinect dapat mengambil informasi *depth.*
   - Mempelajari bagaimana mengolah informasi *depth.*
   - Mempelajari metode *skeletal tracking* yang telah disediakan oleh Kinect SDK.

- Mempelajari bagaimana cara mengolah informasi dari *skeletal tracking.*
- Mempelajari dasar-dasar dari *backpropagation neural network*

2. **Perancangan Hardware**
   Pada Tugas Akhir ini meliputi *hardware* yang digunakan yaitu kamera kinect sebagai input visual serta data *depth* dan layar monitor sebagai ouput visual. Semua proses pada kamera Kinect dan monitor akan dilakukan oleh bagian *processing unit* berupa laptop.

3. **Perancangan Software**
   Untuk mennggunakan kamera Kinect maka langkah pertama yang dilakukan pada tahap perancangan *software* adalah membuat langkah-langkah program untuk menginisialisasi. Inisialisasi kamera kinect meliputi *RGB imaging, depth imaging,* dan *skeletal tracking. Skeletal tracking* digunakan untuk menentukan titik pengukuran yang akan dicari nilai *depth*-nya. Data -data *depth* ini nantinya akan digunakan sebagai *data sets* yang akan dilatihkan dengan *Backpropagation Neural Network* untuk menentukan ukuran lebar badan pria. Dan berdasarkan ukuran tersebut nantinya akan di masukkan kedalam katergori ukuran baju yang ada

4. **Pengujian Sistem**
   Pengujian Sistem dilakukan untuk memperoleh tingkat ketelitian dari sistem yang telah dirancang. Pengujian dilakukan secara bertahap yaitu pengujian untuk menentukan estimasi jarak antar bahu dan estimasi lebar badan pada subjek pelatihan dan subjek yang tidak dilatihkan pada jarak 1500 mm. Pengujian berikutnya dilakukan dengan memberikan variasi jarak yaitu, 1200mm, 1500mm dan 1700mm.

5. **Penulisan Laporan Akhir**
   Tahap penulisan laporan Tugas Akhir dilakukan pada saat tahap pengujian sistem dimulai serta setelahnya.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

- **Bab 1 : Pendahuluan**
  Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, sistematika penulisan, metodologi, dan relevansi.
- **Bab 2 : Dasar Teori**
  Bab ini menjelaskan tentang teori-teori dasar yang menunjang dalam pengerjaan tugas akhir ini. Teori yang digunakan meliputi teori dasar pengambilan citra RGB, pengambilan citra kedalaman (*depth)*, konversi data *depth*, *skeletal tracking, Backpropagation Neural Network*
- **Bab 3: Perancangan Sistem**
  Bab ini menjelaskan mengenai cara kerja sistem yang meliputi *hardware* maupun *software* untuk melakukan pengukuran badan pria dan memberikan rekomendasi ukuran pakaian berdasarkan hasil pengukuran yang telah didapatkan melalui *image processing* serta *Backprpagtion Neural Network*
- **Bab 4 : Pengujian dan Analisis**
  Bab ini menjelaskan mengenai data hasil pengujian yang disertai dengan analisa.
- **Bab 5 : Penutup**
  Bab ini menjelaskan mengenai kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran yang menunjang untuk pengembangan dimasa yang akan datang.

1.7 Relevansi

Mata kuliah yang mendukung tugas akhir ini adalah Dasar Interaksi Manusia Mesin dan Dasar Elektronika Cerdas. Serta beberapa refrensi mengenai kamera Kinect. Hasil dari Tugas Akhir ini diharapkan dapat dikembangkan lagi untuk ditingkatkan akurasi dan kehandalannya. Aplikasi dari sistem ini dapat diterapkan pada toko baju baik secara langsung maupun toko baju *online*.

---Halaman ini sengaja dikosongkan---

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

Tinjauan pustaka dalam bab ini menjelaskan mengenai sistem-sistem yang berhubungan tugas akhir ini serta telah diimplementasikan oleh penulis-penulis lain. Sedangkan dasar teori menjelaskan tentang teori yang mendukung keseluruhan sistem pada tugas akhir ini.

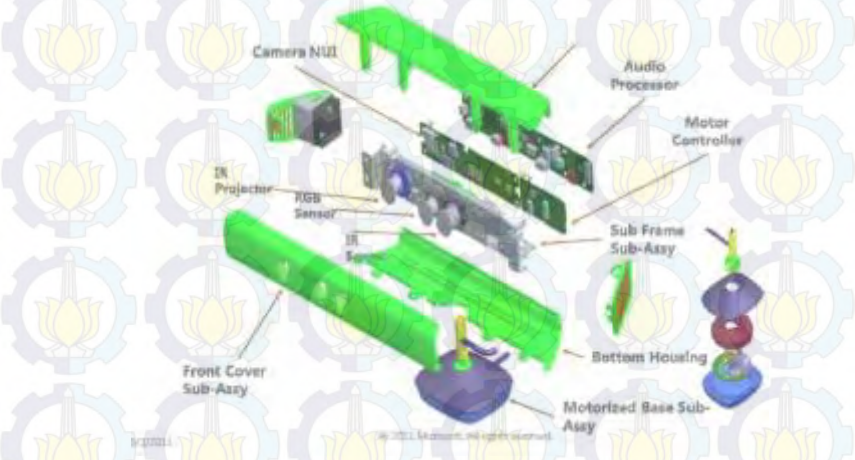
2.1 Antropometri

Antropometri dapat diartikan sebagai ilmu yang berkaitan dengan pengukuran ukuran manusia seperti berat dan proporsi badan[4]. Antropometeri memiliki peranan yang penting pada industri pakaian, ergonomis, dan arsitektur dimana data statistik mengenai distribusi dimensi tubuh pada suatu lingkungan populasi yang akan digunakan untuk mengoptimalkan suatu produk. Antropometri ini dipengaruhi oleh gaya hidup, nutrisi serta etnik dari suatu populasi, sehingga perlu adanya pembaruan secara berkala. Salah satu data antropometri yang digunakan dalam industri pakaian yaitu *chest breadth. Chest breadth* seperti yang terlihat pada gambar 2.1 merupakan lebar badan yang diukur dari ujung ketiak ke ujung ketiak lainnya[5]. Kegunaan dari *chest breadth* dalam dunia industri pakaian yaitu untuk menentukan lebar ukuran baju yang diproduksinya.

Gambar 2.1 *Pengukuran* chest breadth *oleh NASA*[5]

2.2 Kinect

Kinect merupakan seperangkat media yang digunakan oleh Microsoft untuk mendeteksi gerakan pada Xbox 360, Xbox One, dan komputer Windows. Dengan menggunakan Kinect user dapat berinteraksi secara realtime dengan perangkat-perangkat tersebut tanpa pengontrol lain. Interaksi tersebut dapat dilakukan melalui Natural User Interface (NUI)[6]. Sedangkan NUI merupakan sebuah interface yang berusaha menjadikan interaksi antara pengguna dengan suatu perangkat teknologi semudah mungkin sehingga terasa natural. Pada Kinect teradapat dua buah kamera, yaitu kamera RGB dan *infra red depth finding camera*. Selain kamera, Kinect juga dilengkapi dengan multi-array microphone.



Gambar 2.2 Bagian dari kamera Kinect[6]

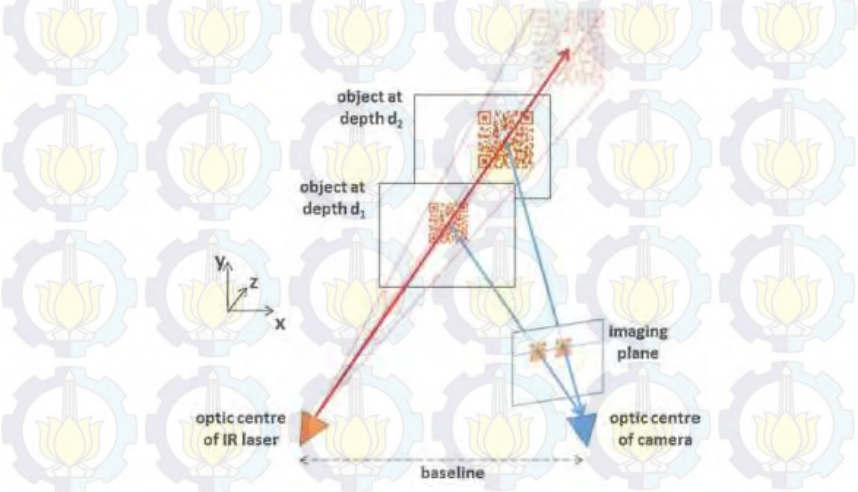
Kamera RGB pada Kinect mempunyai resolusi pengaturan dasar resolusi sebesar 640 x 480 dengan kecepatan 30 *frame* per detik dan kemampuan maksimal dari kamera RGB Kinect yaitu menghasilkan resolusi sebesar 1280 x 1024 dengan *frame* per detik yang lebih lambat. Sedangkan untuk *infrared depth finding camera* terdiri dari dua bagian utama yaitu *IR laser projector* dan *IR camera* yang dapat menghasilkan resolusi maksimal sebesar 640 x 480 dengan kecepatan 30 *frame* per detik. *IR laser projector* akan disebarkan secara acak yang pada akhir-nya akan mengenai suatu objek dan memantulkan cahaya inframerah. Pantulan

tersebut ditangkap oleh *IR camera* yang selanjutnya digunakan untuk pengolahan data kedalaman (*depth)* suatu objek pada suatu *frame*[7].

2.2.1 Citra Kedalaman *(Depth Imaging)*

Untuk memperoleh sebuah citra kedalaman maka diperlukan data kedalaman *(depth)*. Dalam kamera Kinect, data kedalaman diperoleh dari proses pemancaran dan penangkapan cahaya inframerah dari *Kinect Depth Sensors*. Posisi geometri antara pemancar inframerah (*IR* projector) dengan penangkap inframerah serta pola titik dari inframerah telah diketahui[8]. Jika titik inframerah yang dipancarkan dapat dicocokkan dengan titik yang ada pada citra yang sedang diamati maka data *depth* akan didapatkan.

Data *depth* yang didapatkan dapat ditampilkan menjadi suatu citra kedalaman *(Depth Image)*. Pada umumnya *depth image* menampilkan perubahan intensitas warna. Dimana perubahannya sesuai dengan jarak antara objek dengan kamera. Pemilihan warna yang digunakan untuk menampilkan *depth image* mulai dari putih sampai hitam, tergantung dari jarak objek terhadap Kinect. Contohya untuk jarak objek yang terlalu dekat dengan Kinect akan berwarna putih atau lebih terang, sedangkan objek yang jauh dengan akan berwarna gelap. Tidak semua objek yang tertangkap oleh Kinect mempunyai nilai



Gambar 2.3 Pengambilan data *depth*[8]

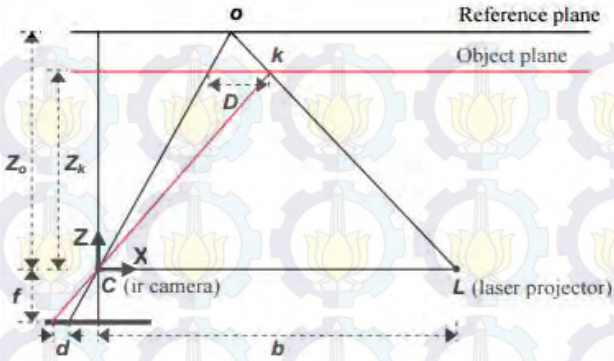
Gambar 2.4 *Depth image* dan rgb *image* pada Kinect

depth, hal ini dikarenakan *depth sensors* pada kinect mempunyai jarak minimum dan jarak maksimum untuk mendapatkan data *depth*. Karena *depths sensors* pada kinect menggunakan pantulan sinar inframerah, maka objek yang tidak dapat memantulkan sinar inframerah tidak akan memiliki nilai *depth* dan akan berwarna hitam. Hasil dari data-data *depth* tersebut selanjutnya di representasikan pada sebuah citra yang ada pada gambar 2.4

2.2.2 Depth Map

Dalam komputer grafis 3 dimensi, *depth map* adalah sebuah gambar atau *image channel* yang berisikan informasi jarak suatu permukaan suatu objek dari sebuah sudut pandang tertentu. Informasi jarak pada *depth map* umumnya dinyatakan dengan koordinat Z. Koordinat Z pada *depth map* tidak dapat disamakan dengan koordinat Z dunia, karena koordinat Z pada *depth map* bersifat relatif terhadap sudut pandang kamera.

Metode yang digunakan pada Kinect untuk menentukan estimasi jarak (*depth data*) yaitu proses tringaulasi [9]. Seperti pada gambar 2.5 yang menunjukan hubungan antara jarak dari suatu titik objek *k* ke sensor relatif terhadap bidang referensi diukur *disparity d*. Untuk mendapatkan sistem koordinat 3 dimensi, di asumsikan origin dari sistem koordinat *depth* berada pada pusat perspektif dari kamera inframerah. Dimana sumbu Z ortogonal terhadap *image plane* menuju objek. Sumbu X tegak lurus terhadap sumbu Z pada arah baseline *b* amtara kamera inframerah dan pemancar inframerah, sedangkan sumbu Y tegak lurus terhadap X dan Z dan menjadikan aturan koordinat tangan kanan.



Gambar 2.5 Representasi hubungan *depth-disparity*[10]

Koordinat *image plane* dengan *depth data* dimodelkan menjadi seperti berikut ini[10].

$$Z_k = \frac{Z_o}{1+\frac{Z_o}{fb}d} \quad (2\text{-}1)$$

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x) \quad (2\text{-}2)$$

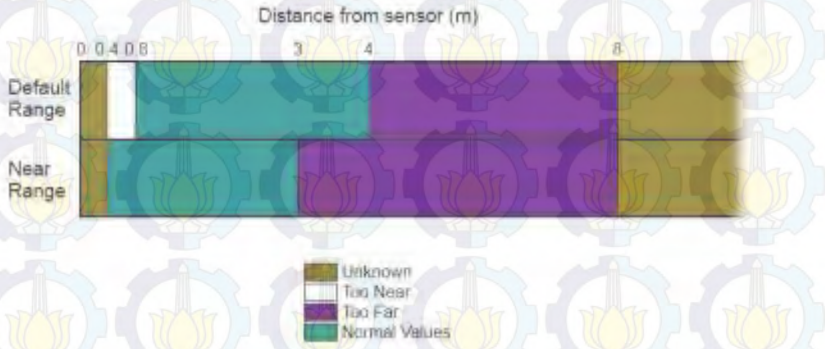
$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y) \quad (2\text{-}3)$$

Pada persamaan (2-1) Z_k merupakan penurunan model matematika untuk *depth* dari *disparity* yang diamati dengan parameter konstan jarak dari referensi pola inframerah (Z_o), lebar fokus (*f*), *base length* (*b*), dan *disparity* yang terukur (*d*). Dimana Z_o, *f*, dan *d* dapat ditentukan melalui kalibrasi. Sedangkan pada persamaan (2-2) dan (2-3) X_k dan Y_k merupakan koordinat citra dari titik *principal offsets* x_o dan y_o, sedangkan δ_x dan δ_y merupakan koefisien distorsi lensa[10].

2.2.3 Depth Space Range

Depth space range merupakan sebuah rentang jarak yang digunakan Kinect untuk mengukur *depth* dari suatu objek. Kinect mempunyai 2 jenis mode untuk melakukan pengukuran *depth*. Mode tersebut yaitu *deffault range* dan *near range*. Pada mode *deffault* maka rentang jarak pengukuran yang efektif yaitu antara 0,8 meter hingga 4

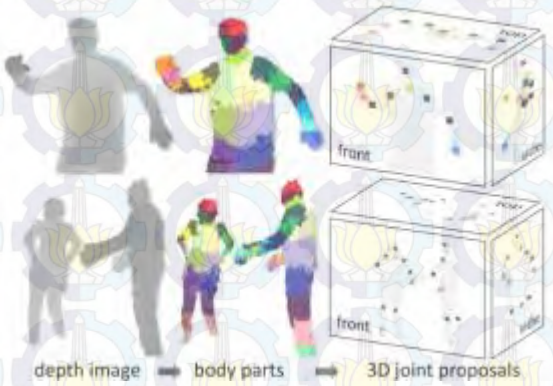
meter. Sedangkan pada mode *near* rentang jarak pengukuran yang efektif yaitu anatar 0,4 meter hingga 3 meter[6].



Gambar 2.6 Kinect *depth space range*[6]

2.2.4 Skeletal Tracking

Natural User Interface akan menjadi lebih mudah dan berjalan lebih lancar dengan bantuan *skeletal tracking*. Dalam *skeletal tracking* pada Kinect SDK 1.8, ia dapat menampilkan maksimal 2 orang dengan maksimal 20 sendi untuk setiap orangnya dan dapat mengenali 6 orang sekaligus. Sendi ini lah yang dipilih menjadi fitur untuk menentukan lebar badan yang akan di perkirakan.



Gambar 2.7 Proses dari *skeletal tracking*[11]

Skeletal tracking dihasilkan melalui dua tahapan yang utama yaitu melalui *depth image* setiap bagian badan diberikan disegmentasi. Segmentasi ini dibagikan berdasarkan posisi sendi. Segmentasi tersebut dilakukan dengan *randomized decision forest* yang akan melakukan pembelajaran dari *depth image* dengan bagian *skeletal* yang sebelumnya telah diketahui[11]. Setelah bagian tubuh telah berhasil disegmentasi langkah berikutnya yaitu memberikan estimasi dimana posisi sendi berada. Penentuan posisi sendi dilakukan dengan algoritma *mean shift* untuk mendapatkan hasil distribusi probabilitas yang kuat (*robust*).

Mode yang terdapat pada *skeletal tracking* yaitu ada 2, yang pertama yaitu *deffault mode*. Mode pertama ini dapt mengenali 20 sendi yang ada pada manusia dan hasilnya akan optimal jika penggunanya berdiri. Sedangkan yang kedua yaitu *seated mode* yang dapat melakukan *tracking* 10 sendi. Untuk jenis sendi yang ada pada masing-masing mode dapat dilihat pada tabel 2.1[6].

Tabel 2.1 Perbandingan sendi antara *default mode* dan *seated mode*

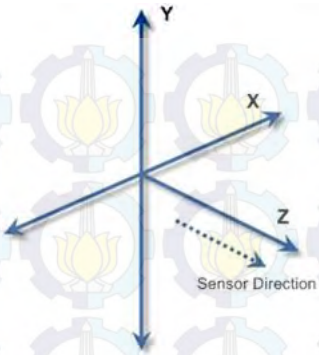
No.	Sendi	Deffault Mode	Seated Mode
1	Hip Center	Ada	Tidak
2	Spine	Ada	Tidak
3	Shoulder Center	Ada	Ada
4	Head	Ada	Ada
5	Shoulder Left	Ada	Ada
6	Elbow Left	Ada	Ada
7	Wrist Left	Ada	Ada
8	Hand Left	Ada	Ada
9	Shoulder Right	Ada	Ada
10	Elbow Right	Ada	Ada
11	Wrist Right	Ada	Ada
12	Hand Right	Ada	Ada
13	Hip Left	Ada	Tidak

No.	Sendi	Default Mode	Seated Mode
14	Knee Left	Ada	Tidak
15	Ankle Left	Ada	Tidak
16	Foot Left	Ada	Tidak
17	Hip Right	Ada	Tidak
18	Knee Right	Ada	Tidak
19	Ankle Right	Ada	Tidak
20	Foot Right	Ada	Tidak

Pada *frame* yang ada *skeleton* dapat mempunyai dua macam kondisi *tracking*. Kondisi yang pertama yakni *tracked*, pada kondisi ini *skeleton* yang ada akan memberikan informasi posisi koordinat sendi yang ada. Sedangkan kondisi yang kedua yaitu *position only* merupakan kondisi dimana *frame* akan memberikan informasi mengenai posisi dari penggunanya, namun tidak ada informasi koordinat dari sendi

2.2.5 Sekeleton Space

Ketika kondisi *tracking skeleton* berada dalam kondisi *tracked* maupun *position only* maka setiap *frame*, *depth image* akan ditangkap dan diproses oleh Kinect *runtime* menjadi *skeleton data*. *Skeleton data* berisikan posisi tiga dimensi untuk *skeleton* manusia. Pada *skeleton space* ini, posisi koordinat ditunjukkan oleh sumbu garis x,y, dan z.[6] Tetapi *depth space* dan *skeleton space* tidak bisa disamakan. Hal ini karena titik origin mereka tidaklah sama. Bila dilihat pada layar, *skeleton space* mempunyai origin (0,0,0) yang terletak tepat ditengah layar. Sedangkan *depth space* mempunyai origin (0,0,0) yang terletak pada pojok kiri atas layar. Selain itu perbedaanya sumbu z pada *skeleton space* di definisikan dengan standart satuan meter berbeda dengan *depth space* dimana sumbu z didefinisikan dengan standart satuan milimeter. Koordinat yang digunakan pada *skeleton space* menggunakan aturan tangan kanan dimana sumbu z positif memanjang seiring dengan arah sensor Kinect. Sumbu y positif memanjang ke atas dan sumbu positif x memanjang ke arah kanan seperti yang terlihat pada gambar 2.8[6].



Gambar 2.8 *Skeleton space*[6]

2.3 Artificial Neural Networks

Menurut Haykin, S *Artificial Neural Network* adalah sebuah prosesor paralel yang tersdistribusi secara masif yang secara natural mempunyai kecenderungan untuk menyimpan informasi pengalaman yang nantinya dapat digunakan lagi[12]. Hal tersebut menyerupai seperti otak dalam dua aspek yaitu:

a. Informasi pengalaman tersebut didapatkan melalui proses pembelajaran.
b. Kekuatan antara koneksi *interneuron* dikenal sebagai bobot sinaptik yang digunakan untuk menyimpan informasi pengalaman.

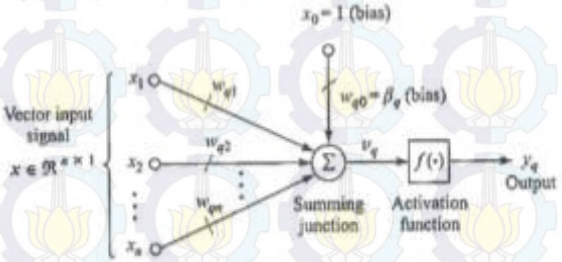
Artificial neural networks atau Jaringan Saraf Tiruan (JST) merupakan sebuah metode pembelajaran berbasis stiatistik. Metode ini terinspirasi dari jaringan saraf manusia dimana otak sebagai unit pengolah yang utama. Pada jaringan saraf manusia, setiap saraf terhubung dengan saraf lainnya melalui sinapsis sedangkan pada jaringan saraf tiruan setiap saraf diibaratkan sebuah *node* yang saling terhubung dengan *node* lainnya melalui sebuah garis yang memiliki bobot nilai. Bobot nilai dimaksudkan seperti pengalaman manusia dalam menerima rangsangan maupun pembelajaran yang diulang secara terus menerus hingga akhirnya manusia dapat mengerti

Secara sederhana jaringan saraf tiruan ini akan belajar secara terus menerus sesuai dengan tujuan yang telah ditentukan sebelumnya.

2.3.1 Struktur Neural Networks [13]

Dasar struktur dari jaringan saraf tiruan dapat dilihat pada gambar 2.9. Terdapat 3 komponen dasar pada jaringan saraf tiruan yaitu:

a. Terdapat beberapa set dari sinapsis yang saling terkait dengan *synaptic weights*, dimana komponen vektor x_j dengan j = 0, 1, 2, . . . , n. Setiap vektor x_j menjadi masukkan ke sinapsis j dan terhubung dengan neuron q melalui *synaptic weight* w_{qj}. Penulisan *subscript* yang pertama memiliki keterkaitan *neuron* tertentu, sedangkan *subscript kedua* memiliki keterkaitan dengan elemen input vektor.
b. *Device* penjumlah berfungsi untuk menjumlahkan semua sinyal yang menuju pada *device* penjumlah, dimana setiap masukkan dikali dengan *synaptic weight dan bias* (khusus x_0) yang bersangkutan lalu dijumlahkan. Semua operasi ini terjadi pada penjumlah v_q dan masih merupkan operasi linier.
c. Fungsi aktivasi $f(\cdot)$ akan membatasi amplitudo dari keluaran neuron y_q, bila menggunakan fungsi aktivasi nonlinier.



Gambar 2.9 Struktur dasar *Neural Networks*

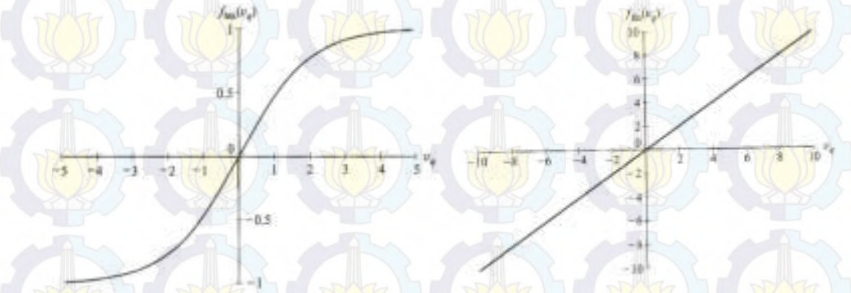
Secara matematis, operasi pada *Neural Networks* dalam gambar 2.9 dapat dituliskan menjadi persamaan berikut:

$$v_q = \sum_{j=0}^{n} w_{qj}\, x_j \qquad (2\text{-}4)$$

$$y_q = f(v_q) \qquad (2\text{-}5)$$

2.3.2 Fungsi Aktivasi [13]

Fungsi aktivasi terdiri dari berbagai macam jenis seperti fungsi *unit step,* linier, trigonometeri, hiperbolik, dan sigmoid. Pada metode *backporpagation* kombinasi dari dua atau lebih fungsi aktivasi akan menentukan penggunaan dari *Neural Networks*, contohnya kombinasi fungsi aktivasi linier dan hiperbolik tangen sigmoid (bipolar sigmoid) menghasilkan regresi dari masukan-masukan yang diberikan.



(a) (b)

Gambar 2.10 (a) Fungsi Aktivasi hiperbolik tangen sigmoid dan fungsi aktivasi linier (b)

Fungsi aktivasi bipolar sigmoid memiliki rentang keluaran antara -1 hingga -1. Sedangkan untuk fungsi aktivasi linier ia tidak memiliki rentang keluaran. Fungsi aktivasi bipolar sigmoid memiliki persamaan (2-6) dengan turunan (2-7) sedangkan fungsi aktivasi linier memiliki persamaan (2-8).

$$y_q = f_{hts}(v_q) = \frac{1 - e^{-2v_q}}{1 + e^{-2v_q}} \tag{2-6}$$

$$g_{hts}(v_q) = \left(1 + f_{hts}(v_q)\right)\left(1 + f_{hts}(v_q)\right) \tag{2-7}$$

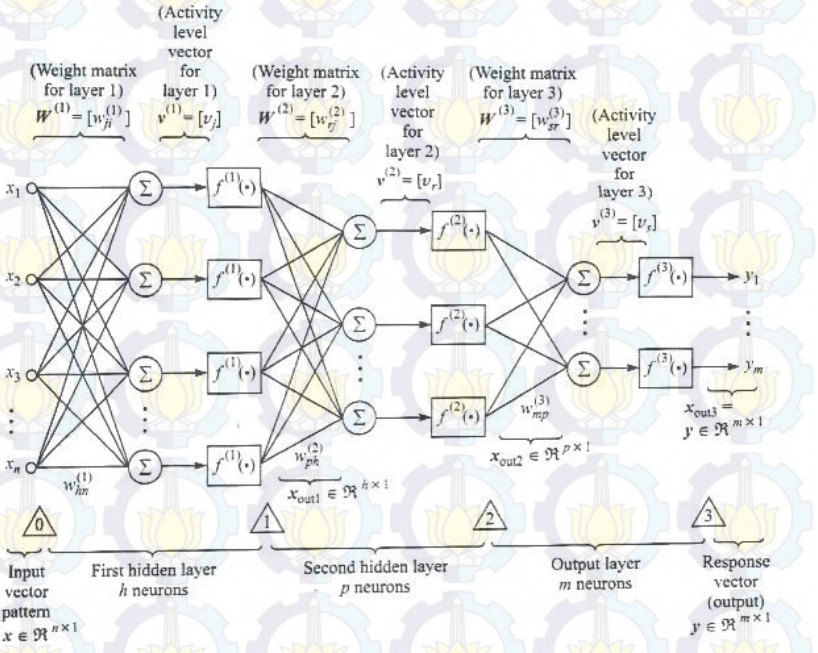
$$y_q = f_{lin}(v_q) = v_q \tag{2-8}$$

2.3.3 Backpropagation Neural Networks [13]

Backpropagation merupakan sebuah metode pembelajaran dalam *Multi Layer Perceptron* pada *Neural Networks.* Selama proses pembelajaran, *weight* dan bias diatur sedemikian rupa, untuk memperoleh

error yang minimum. Prinsip dasar dari algoritma *backpropagation* adalah *memperbaiki* bobot-bobot jaringan sehingga keluaran yang dihasilkan sesuai dengan batasan error minimal yang ditentukan.

Pada gambar 2.11, dimana i = 1, 2, . . . , n; j = 1, 2, . . . ,h; r = 1, 2, . . . , p; s = 1, 2, . . . , m; $f(\cdot)^{(1)}$ adalah fungsi aktivasi pada layer pertama, $f(\cdot)^{(2)}$ adalah fungsi aktivasi pada layer kedua, dan $f(\cdot)^{(3)}$ adalah fungsi aktivasi pada layer ketiga. Diasumsikan terdapat bias pada setiap neuron yang berada pada setiap layer dan juga terdapat matriks bobot yang terhubung antara *layer* berikutnya dengan *layer* setelahnya, yaitu $\boldsymbol{W}^{(\ell)}$ dengan $\ell = 1, 2, 3$. *Output* pada *layer* pertama, kedua, dan ketiga terdapat pada persamaan (2-9), (2-10) dan (2-11). Sedangkan respon akhir dari jringan (2-12) merupakan hasil dari subtitusi persamaan (2-9), (2-10) dan (2-11).



Gambar 2.11 Arsitektur tiga *layer perceptron*

$$\boldsymbol{x}_{out1} = \boldsymbol{f}^{(1)}[\boldsymbol{v}^{(1)}] = \boldsymbol{f}^{(1)}[\boldsymbol{W}^{(1)}\,\boldsymbol{x}] \quad (2\text{-}9)$$

$$\boldsymbol{x}_{out2} = \boldsymbol{f}^{(2)}[\boldsymbol{v}^{(2)}] = \boldsymbol{f}^{(2)}[\boldsymbol{W}^{(2)}\,\boldsymbol{x}_{out2}] \quad (2\text{-}10)$$

$$\boldsymbol{y} = \boldsymbol{x}_{out3} = \boldsymbol{f}^{(3)}[\boldsymbol{v}^{(3)}] = \boldsymbol{f}^{(3)}[\boldsymbol{W}^{(3)}\,\boldsymbol{x}_{out3}] \quad (2\text{-}11)$$

$$\boldsymbol{y} = \boldsymbol{f}^{(3)}[\boldsymbol{W}^{(3)}\boldsymbol{f}^{(2)}[\boldsymbol{W}^{(2)}\boldsymbol{f}^{(1)}[\boldsymbol{W}^{(1)}\,\boldsymbol{x}]]] \quad (2\text{-}12)$$

Pembelajaran dari model *neural networks* dengan metode *backpropagation* dimulai ketika *output* yang dihasilkan oleh *feedforward* tidak sesuai dengan yang *output* yang diharapkan. Perbandingan *output* ini didapatkan dengan cara menghitung error pada *layer output* menggunakan persamaan (2-13) dan pada se mua *layer* dengan persamaan (2-14). Setelah itu bobot nilai dan bias pada jaringan saraf tiruan akan diperbaharui dengan konstanta *learning rate* (μ) berdasarkan persamaan (2-15). 0

$$\delta_j^{(\ell)} = \left(d_{qh} - x_{out,j}^{(\ell)}\right) g\left(v_j^{(\ell)}\right) \quad (2\text{-}13)$$

$$\delta_j^{(\ell)} = \left(\sum_{h=1}^{\ell+1} \delta_h^{(s+1)}\, w_{hj}^{(\ell+1)}\right) g\left(v_j^{(\ell)}\right) \quad (2\text{-}14)$$

$$w_{ji}^{(\ell)}(k+1) = w_{ji}^{(\ell)}(k) + \mu\, \delta_j^{(\ell)}\, x_{out,j}^{(\ell-1)} \quad (2\text{-}15)$$

2.4 Euclidien distance[14]

Euclidiean distance merupakan sebuah dasar dari penentuan sebuah posisi atau lokasi relatif dari sebuah informasi jarak pada sebuah titik tertentu. Dimisalkan jarak antara titik p dan q pada bidang dimensi n mempunyai persamaan *euclidean distance* seperti berikut ini.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} \quad (2\text{-}16)$$

2.5 Kinect SDK [6]

Kinect Software Development Kit (SDK) merupakan sebuah *library* yang digunakan untuk melakukan pengembangan terhadap kamera Kinect. Sejak tahun 2011 SDK ini sudah mendukung sistem operasi Windows 7 dan kompatibel dengan *driver* perangkat Kinect.

Dengan menggunakan SDK, para pengembang dapat merancang berbagai macam jenis aplikasi dengan C++, C# atau Visual Basic menggunakan Visual Studio 2010. Kinect SDK juga mendukung kebutuhan untuk *image processing* seperti melakukan pengambilan, pengolahan serta penampilan data gambar. Interaksi antara manusia dan mesin melalui NUI juga dapat dilakukan. Pada umumnya SDK digunakan untuk pembuatan sebuah *control* pada suatu *game* yang berbasis Kinect.

2.6 Microsoft Visual Studio [15]

Microsoft Visual Studio merupakan sebuah *Integrated Development Environment* (IDE) yang dikembangkan oleh Microsoft. IDE ini digunakan untuk mengembankan atau membuat program-program komputer. *Platforms* yang digunakan pada Visual Studio yakni Windows API, Windows Forms, Windows Presentation Foundation, dan yang lainnya. Dengan adanya *Intellisense* yaitu sebuah editor kode yang dapat melengkapi kode, maka pengguna dapat dengan mudah dan cepat dalam menuliskan program yang akan dibuatnya. Selain pembuatan program yang mudah, Visual Studio menyediakan proses *debug* pada *souce-level* dan juga pada *machine-level*. Hal ini membuat proses *debug* menjadi lebih mudah. Bahasa pemrogaman yang didukung oleh Visual Studio diantaranya C, C++, C++/CLI, VB.Net, dan C#.

2.7 OpenCV [16]

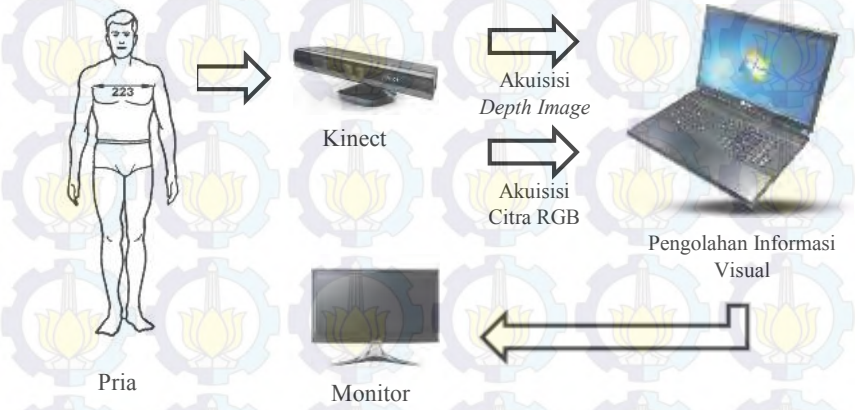
Open Source Computer Vision (Open CV) merupakan sebuah *library* program yang dikususkan untuk pemrogaman *computer vision* secara *real-time*. OpenCV dikembangkan oleh Intel dan saat ini ia didukung oleh Willow Garage. OpenCV dibuat dalam C++ sehingga penggunaan utamanya menggunakan bahasa C++, tetapi sekarang OpenCV mendukung *interface* selain C++, seperti Python, Java, C#, MATLAB. Hingga saat ini OpenCV dapat berjalan dengan baik pada sistem operasi Windows, Linux, Mac OS, Android, dan IOS.

2.8 MATLAB [17]

Matrix Laboratory (MATLAB) merupakan sebuah *environment* yang digunakan untuk melakukan penghitungan numerik berbasis matriks. MATLAB dapat melakukan manipulasi matriks, *plotting* suatu fungsi dan data, MATLAB dilengkapi dengan *interface* C, C++, Java, Fortran, dan Python. Untuk memudahkan penggunanya, MATLAB menyediakan berbagai macam jenis *toolbox* diantaranya *Curve Fitting Toolbox*, *Neural Network Toolbox*, *Control System Toolbox*, dan lain-lain.

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan pembuatan sistem perangkat lunak untuk melakukan pengolahan informasi visual. Sistem pengukuran lebar badan pria terdiri dari 4 komponen yang saling berhubungan diantaranya yaitu seorang pengguna, kamera Kinect, *processing unit*, pengolahan informasi visual, dan media komunikasi untuk pengguna. Hubungan antara komponen-komponen dapat di ilustrasikan pada gambar 3.1 berikut.



Gambar 3.1 Ilustrasi cara kerja sistem

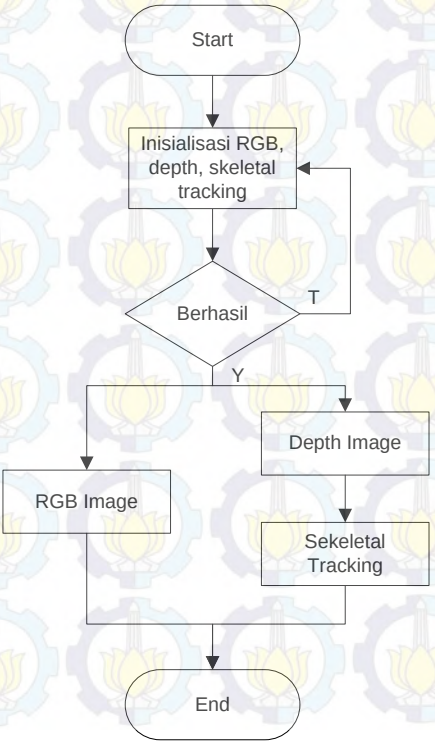
3.1 Prinsip Kerja

Menurut ilustrasi pada gambar 3.1, prinsip kerja dari sisten pengukuran lebar badan pria yang akan dirancang dimulai dari pengambilan informasi visual. Kinect merupakan perangkat yang digunakan untuk mengambil informasi visual dari pengguna. Informasi visual yang diambil diantaranya yaitu citra RGB dan *depth image.* Selanjutnya informasi visual tersebut akan dikirimkan menuju *processing unit*. Pada sistem yang akan dirancang, sebuah *notebook* digunakan sebagai *processing unit*. Informasi visual akan diolah untuk mendapatkan estimasi lebar badan dari pengguna. Dan terdapat program *inteface* yang akan membantu pengguna berinteraksi dengan sistem yang akan dirancang.

3.2 Kinect

Kamera Kinect merupakan perangkat yang digunakan untuk mengakuisisi citra RGB dan *depth image*, berikut ini merupakan spesifikasi dari kamera Kinect yang digunakan:

- Color Camera : 640 x 480 @ 30 fps
- Depth Camera : 320 x 240 @ 30 fps
- Jarak *Depth* Maksimal : 4 Meter
- Jarak *Depth Minimal* : 80 Sentimeter
- Jangkauan Pandangan Horizontal : 57 Derajat
- Jangkauan Pandangan Vertikal : 43 Derajat
- Sendi *Skeleton* yang disediakan : 20
- Jumlah *Skeleton Tracked* : 2



Gambar 3.2 Diagram alir inisialisasi kinect

Inisialisasi merupakan langkah awal dalam penggunaan kamera Kinect. Inisialisasi ini menentukan sensor-sensor yang akan digunakan kamera Kinect untuk melakukan pengambilan data. Disini penulis akan melakukanan inisialisasi untuk tiga jenis penggunaan kamera Kinect yaitu *color stream, depth stream,* dan *skeletal tracking*. *Color stream* merupakan *data stream* yang digunakan untuk melakukan pengambilan data citra RGB dengan cara, `NUI_INITIALIZE_FLAG_USES_COLOR` *Depth stream* juga termasuk *data stream* yang digunakan untuk mengambil data, tetapi data yang diambil disini yaitu berupa data kedalaman. Proses inisialisai yaitu `NUI_INITIALIZE_FLAG_USES_DEPTH`. Sedangkan *skeletal tracking* digunakan untuk melakukan identfikasi pada *frame* apakah terdapat data yang menyerupai *skeletal. Flag* yang digunakan yaitu `NUI_INITIALIZE_FLAG_USES_SKELETON`. Untuk menggunakan tiga macam pengambilan data secara sekaligus, maka *flag* inisialisasi ini dapat digabungkan dengan *bitwise* OR sehingga menjadi:

```
hr= m_pNuiSensor -> NuiInitialize (NUI_INITIALIZE_FLAG_USES_COLOR|
NUI_INITIALIZE_FLAG_USES_DEPTH | NUI_INITIALIZE_FLAG_USES_SKELETON
);
```

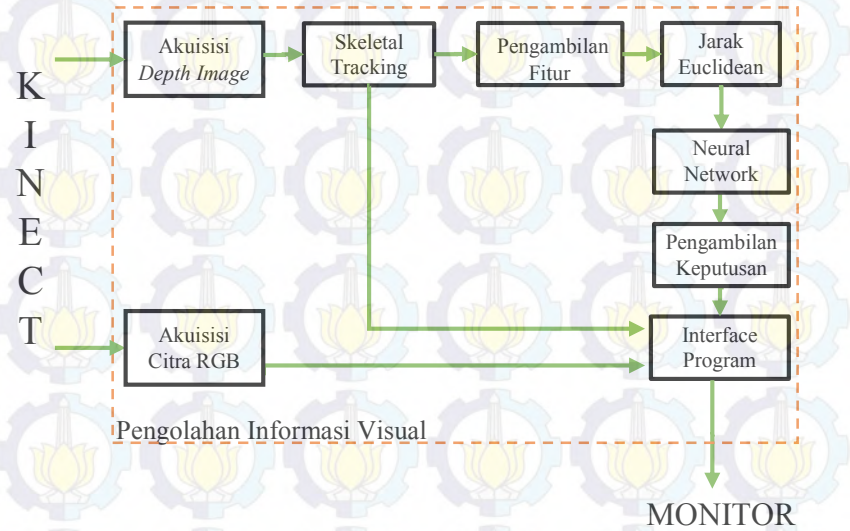
3.3 Pengolahan Informasi Visual

Untuk menentukan estimasi lebar badan dari penggunanya, maka sistem perlu mengambil informasi-informasi visual yang berkaitan dengan penggunanya. Langkah-langkah sistem untuk melakukan estimasi lebar badan pengguna menurut gambar 3.2 adalah seperti berikut:

a. Kinect menangkap informasi citra RGB, data *depth* setiap *frame*. Bila ada orang pada *depth frame,* maka Kinect akan melakukan *skeletal tracking*.
b. Citra RGB dan *skeletal traking* akan diperlihatkan ke *user* melalui program *interface.*
c. *Skeletal tracking* akan memberikan posisi setiap sendi dalam koordinat *skeleton space*.
d. Fitur yang diambil adalah jarak *depth* dari beberapa sendi yang dipilih.
e. Berdasarkan posisi sendi bahu kanan dan bahu kiri, jarak antar bahu dapat diketahui dengan Jarak Euclidean.
f. Data *depth* yang didapat menjadi masukkan untuk jaringan saraf tiruan yang akan menentukan regresi untuk setiap masukkan yang

diberikan yang pada akhirnya dapat memberikan estimasi lebar badan dari pengguna.

g. Pengambilan keputusan akan memberikan saran kategori ukuran baju berdasar hasil estimasi lebar badan.
h. Perkiraan lebar badan yang didapatkan dari proses jaringan saraf tiruan akan disampaikan ke user melalui program *interface*.



Gambar 3.3 Diagram blok pengolahan informasi visual

3.3.1 Akuisisi Depth *Image*

Langkah pertama untuk visualisasi informasi jarak dari *depth image* yaitu dengan cara melakukan penguncian *depth frame* agar data *depth stream* tidak mengalami perubahan saat dilakukan pembacaan. Data *depth stream* akan dimasukkan pada `depthh` melalui `cvSetData(depthh, pBuff, kuncirect.Pitch)`. `depthh` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame depth stream* sedangkan `kuncirect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*. Setelah dilakukan pembacaan data maka akan divisualkan dengan `cvShowImage("Depth Image",depthh)` serta *frame* dibuka untuk pengambilan data berikutnya.

Gambar 3.4 *Depth image*

```
if (kuncirect.Pitch!=0)
{
  int minDepth = (modenear ? NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
    NUI_IMAGE_DEPTH_MINIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
  int maxDepth = (modenear ? NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
    NUI_IMAGE_DEPTH_MAXIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
  BYTE* runwarna = m_depthRGBX;
    const NUI_DEPTH_IMAGE_PIXEL* BuffRC;
    const NUI_DEPTH_IMAGE_PIXEL* BuffLC;
    const NUI_DEPTH_IMAGE_PIXEL* BuffRS;
    const NUI_DEPTH_IMAGE_PIXEL* BuffLS;
    const NUI_DEPTH_IMAGE_PIXEL* BuffBelly;
    const NUI_DEPTH_IMAGE_PIXEL* BuffChest;
    const NUI_DEPTH_IMAGE_PIXEL* RunBuffer = reinterpret_cast <
      const NUI_DEPTH_IMAGE_PIXEL*>(kuncirect.pBits);
    const NUI_DEPTH_IMAGE_PIXEL* EndBuffer = RunBuffer +
      (cDepthWidth*cDepthHeight) -1;
    int selisih = 0;
    selisih = EndBuffer-RunBuffer;
    while (RunBuffer<EndBuffer)
    {
      USHORT depth= RunBuffer->depth;
      BYTE intensitasnya = static_cast<byte>(depth >= minDepth &&
        depth <= maxDepth ? depth % 255:0);
      *(runwarna++) = intensitasnya;
      *(runwarna++) = intensitasnya;
```

```
      *(runwarna++) = intensitasnya;
      ++runwarna;
      ++RunBuffer;
    }
    RunBuffer = RunBuffer-selisih;
    USHORT * pBuff = (USHORT*)m_depthRGBX;
    cvSetData(depthh,pBuff,kuncirect.Pitch);
  }
cvShowImage("Depth Image",depthh);
tekstur->UnlockRect(0);
tekstur->Release();
ReleaseFrame:
m_pNuiSensor-> NuiImageStreamReleaseFrame (m_pDepthStreamHandle,
  &imageFrame);
cvWaitKey(1);
```

3.3.2 Akuisisi Citra RGB

Gambar 3.5 Citra RGB

Untuk menampilkan citra RGB maka *frame color stream* akan dikunci agar saat pengambilan data, data pada *color frame* tidak berubah. Lalu data *color stream* dimasukkan kedalam variabel `color`, dengan cara `cvSetData(color, pBuffer, LockedRect.Pitch)`, dimana `color` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame color stream* sedangkan `LockedRect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*.

```
if (LockedRect.Pitch != 0)
{
```

```
  BYTE * pBuffer = (BYTE*) LockedRect.pBits;
  cvSetData(color, pBuffer, LockedRect.Pitch);
}
cvShowImage("Color Image", color);
tekstur->UnlockRect(0);
m_pNuiSensor->NuiImageStreamReleaseFrame(m_pColorStreamHandle,
  &imageFrame);
cvWaitKey(1);
```

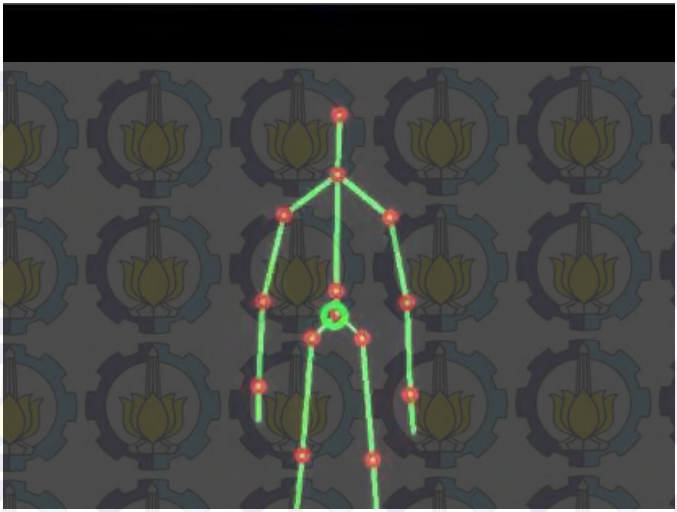
3.3.3 Skeletal Tracking

Pendeteksian *skeletal* memanfaatkan *library* yang berasal dari Kinect. *Library* ini akan mendeteksi kondisi ada atau tidaknya *skeletal* secara. *Skeletal tracking* dimulai dengan memanggil fungsi `void drawSkeleton(IplImage* image)`. Variabel `image` merupakan sebuah data citra. Selanjutnya sensor Kinect akan mendeteksi ada atau tidaknya manusia pada *frame*. Ketika pada *frame* terdeteksi ada manusia maka *skeletal* akan digambarkan secara penuh (20 sendi) pada *frame* dan dapat mengikuti (*tracking*) gerakan manusia.

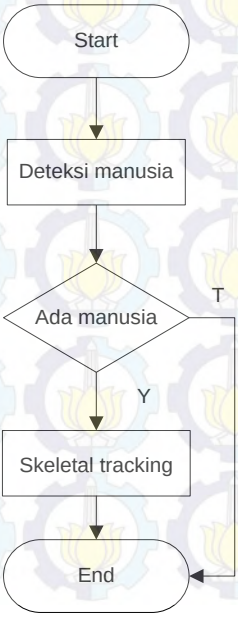
Untuk proses visualisasi dan memudahkan dalam proses berikutnya, koordinat sendi pada *skeletal tracking* akan di konversi dari sistem koordinat *skeleton space* menjadi sistem koordinat *color space*. konversi koordinat dilakukan dengan cara:

```
NuiTransformSkeletonToDepthImage (sendi[jointFrom],
&SenDepthX[jointFrom], &SenDepthY[jointFrom],
NUI_IMAGE_RESOLUTION_640x480);
m_pNuiSensor->
NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(
  NUI_IMAGE_RESOLUTION_640x480, NUI_IMAGE_RESOLUTION_640x480,0,
  (LONG)SenDepthX[jointFrom], (LONG)SenDepthY[jointFrom], 0,
  &SenPosX[jointFrom],&SenPosY[jointFrom]);
```

`NuiTransformSkeletonToDepthImage` akan mengubah posisi sendi *skeleton space* menjadi *depth space* pada resolusi 640x480 dengan nama variabel `SenDepthX[]` untuk sumbu x dan `SenDepthY[]` untuk sumbu y. Setelah itu *depth space* dikonversi menjadi *color space* melalui `NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution` dari resolusi 640x480 dengan resolusi tetap 640x480 dimana variabel `SenPosX[]` untuk sumbu x dan `SenPosY[]`untuk sumbu y.



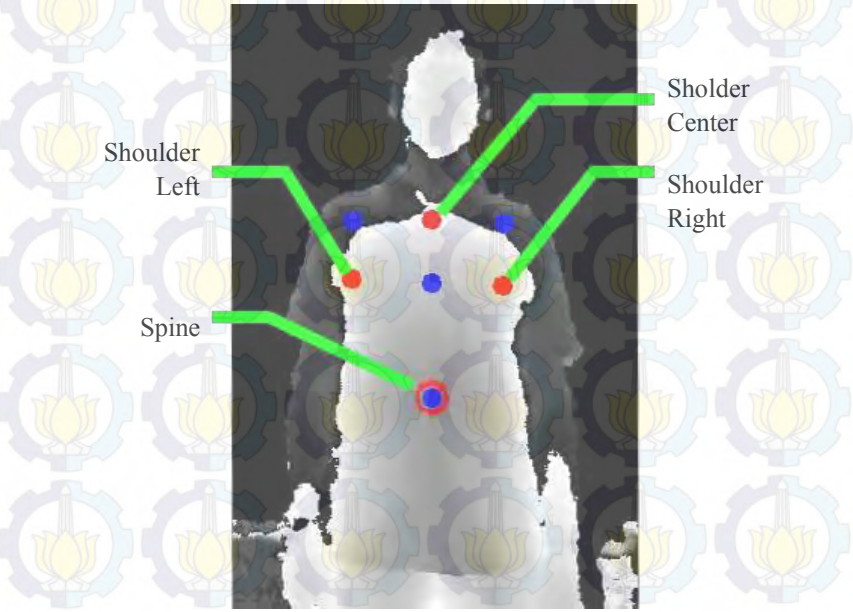
Gambar 3.6 Hasil *Skeletal tracking*



Gambar 3.7 Diagram alir *skeletal tracking*

3.3.4 Pengambilan Fitur

Untuk pengambilan fitur, maka diperlukan beberapa informasi jarak pada *depth image*. informasi tersebut didapat dari empat sendi *skeletal tracking* diantaranya yaitu *spine, shoulder center, shoulder left,* dan *shoulder right*. Bila dilihat pada gambar 3.7 terdapat tiga lingkaran merah, tiga lingkaran biru, dan satu lingkaran biru dengan lingkaran merah disekelilingnya. Tiga lingkaran merah menandakan posisi sendi secara *default* dari kamera Kinect. Tiga sendi ini tidak akan diambil informasi *depth*-nya. Selanjutnya tiga lingkaran biru ini merupakan posisi sendi-sendi bahu dan dada yang mengalami penyesuaian. Pada sendi *shoulder left* dan *right* memakai sumbu Y dari sendi *shoulder center*. Sedangkan pada sendi *shoulder center* memakai sumbu Y dari *shoulder left* dan *right*. Hal ini dilakukan untuk meningkatkan kestabilan dari informasi *depth* yang diambil. Sedangkan satu lingaran biru dengan lingkaran merah disekelilingnya tidak mengalami penyesuaian dan akan diambil informasi *depth*-nya.



Gambar 3.8 Posisi sendi-sendi (*default*) pada *depth image*

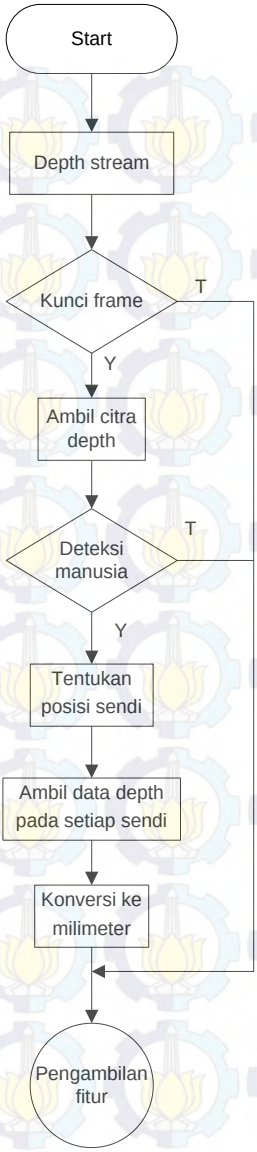
Setelah menentukan posisi sendi yang akan diambil informasi *depth*-nya, konversi data *depth* menjadi milimeter dilakukan untuk mengetahui jarak dari Kinect terhadap permukaan dimana posisi sendi tersebut berada. Untuk mendapatkan jarak antara kamera dengan permukaan posisi sendi tersebut dilakukan saat penguncian *frame depth stream*. Berikut ini cara untuk mendapatkan jarak dalam satuan milimeter untuk setiap posisi sendi:

```
BuffChest = BuffChest + Chestpx;
BuffBelly = BuffBelly + Bellypx;
BuffLS = BuffLS + LSpx;
BuffRS = BuffRS + RSpx;
bellydepth = BuffBelly->depth;
chestdepth = BuffChest->depth;
LSdepth = BuffLS->depth;
RSdepth = BuffRS->depth;
```

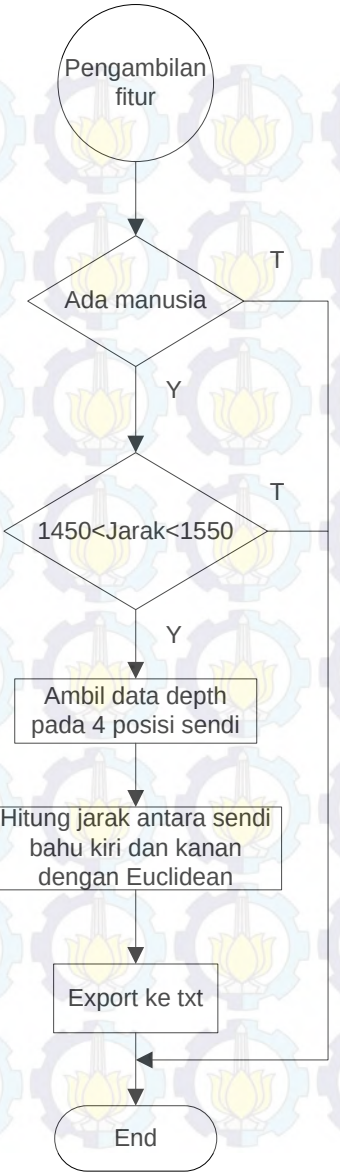
`BuffChest`, `BuffBelly`, `BuffLS`, dan `BuffRS` merupakan variabel piksel yang menyimpan informasi jarak. Sedangkan `Chestpx` (*shoulder center*), `Bellypx` (*spine*), `LSpx` (*shoulder left*), dan `RSpx` (*shoulder right*) merupakan posisi pixel dimana sendi itu berada. Dan hasil pengukuran jarak untuk setiap sendi teradapat pada `bellydepth`, `chestdepth`, `LSdepth`. Dari hasil konversi tersebut, maka fitur-fitur yang digunakan untuk melakukan perkiraan lebar badan akan diproses oleh sistem. Fitur-fitur tersebut diantaranya:

a. Jarak antara Kinect dengan permukaan bahu kiri.
b. Jarak antara Kinect dengan permukaan bahu kanan.
c. Selisih jarak antara permukaan bahu dengan dada.
d. Selisih jarak antara permukaan bahu dengan perut.

Menurut diagram alir pengambilan fitur pada gambar 3.8, langkah pertama untuk pengambilan fitur ini yaitu mendeteksi ada atau tidaknya manusia pada *frame*. Selanjutnya menentukan jarak pengguna dengan kamera akan agar berada pada 1450 mm hingga 1550 mm. Dan yang terakhir menjaga posisi badan lurus dengan kamera dengan cara membandingkan *depth* antara sendi bahu kanan dan bahu kiri sehingga pengambilan fitur dimulai dan di *export* dengan format txt.



Gambar 3.9 Diagram alir pengambilan fitur



Gambar 3.10 Diagram alir pengambilan fitur

3.3.5 Jarak Euclidean

Untuk memperkirakan estimasi lebar badan dari pengguna maka jarak antara sendi pada bahu kanan dan sendi pada bahu kiri akan diestimasi terlebih dahulu. Sendi-sendi yang didapat dari *skeletal tracking* berada dalam koordinat tiga dimensi. Persamaan Euclidean dari titik p ke titik q pada ruang tiga dimensi yaitu:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 - (p_2 - q_2)^2 - (p_3 - q_3)^2} \qquad (3\text{-}1)$$

```
FLOAT diff_x = LS.x - RS.x;
FLOAT diff_y = LS.y - RS.y;
FLOAT diff_z = LS.z - RS.z;
lebar_s = (sqrt(pow(diff_x,2)+pow(diff_y,2)+pow(diff_z,2))*100);
```

3.3.6 Neural Network

Artificial Neural Networks dalam dunia *machine learning* sering disebut dengan *Neural Networks* atau Jaringan Saraf Tiruan (JST) merupakan sebuah metode pembelajaran statistik berdasarkan sistem saraf pada bagian otak. JST dapat melakukan *mapping* antara masukkan dan keluaran ataupun mencari korelasi antar keduanya. Korelasi didapatkan dari pasangan masukkan dan keluaran yang telah dilatihkan.

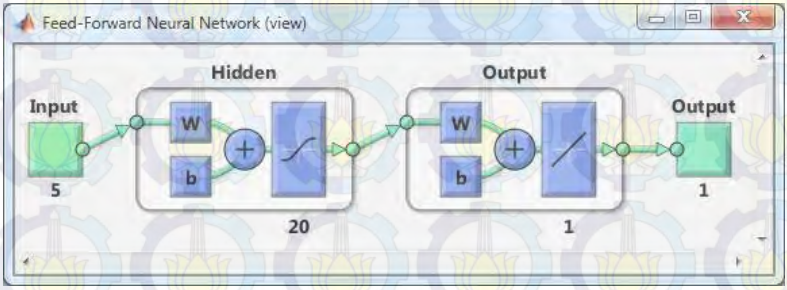
Penggunaan JST karena kemampuannya untuk melakukan generalisasi pada keluaran, meskipun informasi-informasi yang diterima tidak ada pada informasi-informasi yang telah dilatihkan. Selain itu JST juga dapat menentukan tren data yang sangat kompleks yang tidak terlihat oleh pengamatan manusia.

JST sangat membantu bila pada suatu sistem membutuhkan *multiple input* dan juga *multiple output*. Bila sistem menggunakan *single input* dan *single output* maka metode regresi sudah cukup untuk di implementasikan pada sistem. Pada Tugas Akhir ini JST digunakan untuk melakukan pembelajaran agar dapat memperkirakan lebar badan pengguna. Pembelajaran jaringan saraf tiruan menggunakan MATLAB. Penulis menggunakan MATLAB untuk mempercepat proses pembelajaran dan menghindari *overfit* yang dapat terjadi[18]. *Mean Squared Error* (MSE) digunakan untuk mengetahui performa dari jaringan saraf tiruan.

Setelah pengambilan *fitur* di *eksport* dalam bentuk txt, kemudian *file* tersebut akan menjadi sumber data pembelajaran untuk MATLAB.

Jaringan saraf tiruan yang diterapkan disini yaitu *multi layer percetpron backpropagation* menggunakan 5 *node input layer*, 20 *node hidden layer* 1, dan 1 *node output layer*. Fungsi aktivasi yang digunakan yaitu bipolar sigmoid pada *hidden layer* dan fungsi aktivasi linier pada *output layer*. Fungsi aktivasi yang digunakan bertujuan untuk mendapatkan tren data dari estimasi lebar badan. Adapun masukkan yang diberikan yaitu:

- Jarak kedalaman (*depth*) antara permukaan bahu kiri pengguna dengan Kinect.
- Jarak kedalaman (*depth*) antara permukaan bahu kanan pengguna dengan Kinect.
- Selisih kedalaman (*depth*) antara permukaan bahu pengguna dengan permukaan dada pengguna.
- Selisih kedalaman (*depth*) antara permukaan bahu pengguna dengan permukaan perut pengguna.
- Estimasi lebar bahu pengguna berdasarkan metode jarak Euclidean.



Gambar 3.11 Konfigurasi MLP Backpropagation

Setelah MATLAB selesai melakukan pembelajaran maka nilai bobot dan bias antar *node* di *eksport* ke dalam *file* txt dan siap digunakan untuk tahapan estimasi pengukuran lebar badan. Estimasi pengukuran lebar badan atau bagian *feedforward* dari *multi layer backpropagation* di terapkan pada C++ dimulai dengan *import* data-data txt diantaranya:

a. Input.txt, sebagai *input database* yang nantinya akan digunakan untuk proses normalisasi.
b. Output.txt, sebagai *output database* yang nantinya akan digunakan untuk proses normalisasi.

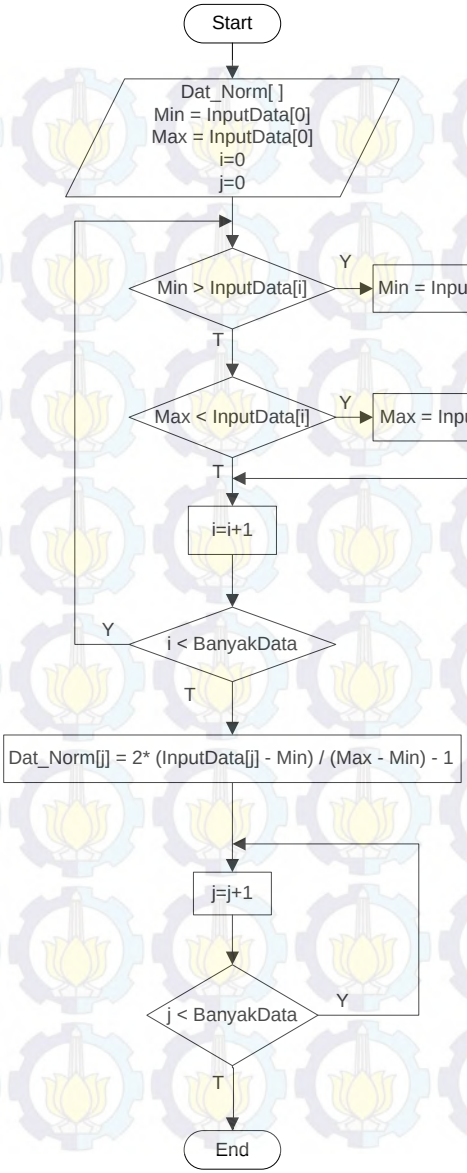
c. w1.txt, bobot nilai setiap node yang berada pada *input layer* dan *hidden layer*.
d. w2.txt, bobot nilai setiap node yang berada pada *hidden layer* dan *output layer*.
e. bias1.txt, nilai bias yang berada pada setiap node pada *hidden layer*.
f. bias2.txt, nilai bias yang berada pada setiap node pada *output layer*.

Untuk meningkatkan performa pebelajaran dari jaringan saraf tiruan, perlu adanya proses normalisasi pada bagian *input*, dan denormalisasi pada bagian *output*. Metode normalisasi (3-2) dan denormalisasi (3-3) yang digunakan yaitu metode min-max. Pada proses normalisasi, nilai minimal dan juga nilai maksimal yang dicari merupakan nilai yang berada pada satu node *input* yang sama, bukan nilai minimal dan maksimal dari lima node *input*.

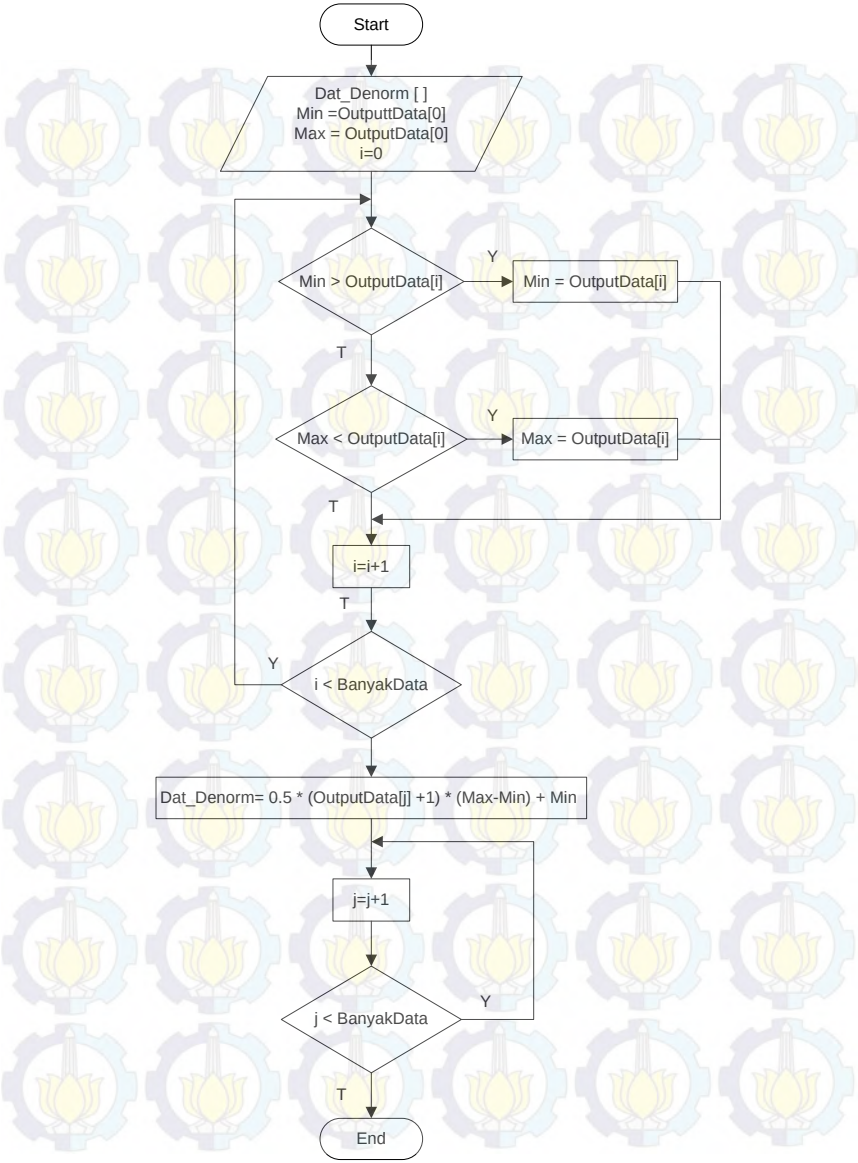
$$y = \frac{(y_{max} - y_{min})(x_{max} - x_{min})}{(x_{max} - x_{min})} - y_{min} \quad (3\text{-}2)$$

$$x = \frac{(x_{max} - x_{min})(y_{max} - y)}{(y_{max} - y_{min})} - x_{min} \quad (3\text{-}3)$$

Dimana:
y_{max} = Nilai maksimum hasil normaliasi
y_{min} = Nilai minimum hasil normaliasi
x_{max} = Nilai maksimum data
x_{min} = Nilai minimum data



Gambar 3.12 Diagram alir normalisasi



Gambar 3.13 Diagram alir denormalisasi

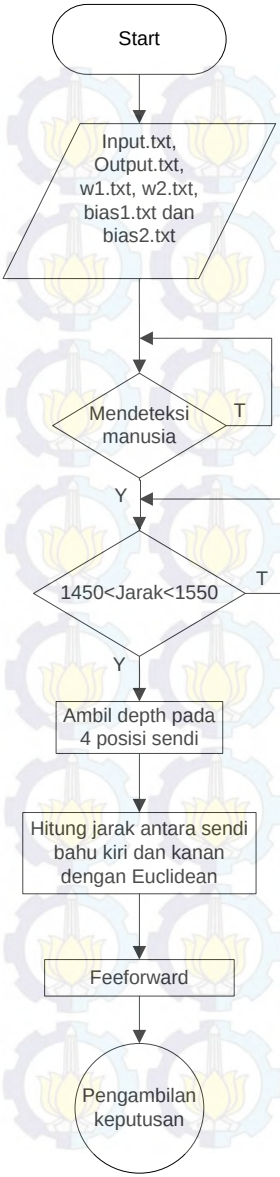
3.3.7 Pengambilan Keputusan

Estimasi lebar badan dilakukan dengan mengambil fitur-fitur dari pengguna yang berdiri di depan kamera dilanjutkan dengan proses *feedforward* MLP. Proses pertama pada *feedforward* yaitu *import* 6 *files* yang telah disediakan, dilanjutkan dengan pengambilan fitur, lakukan normalisasi terhadap fitur, proses dengan *feedforward*, denormalisasi hasil *feedforward* dan yang terakhir lakukan pengambilan keputusan untuk menentukan kategori ukuran baju berdasarkan hasil estimasi lebar badan. Dimana baju dikategorikan menjadi 4 yaitu S, M, L XL dan - (tidak ada ukuran baju yang sesuai), sesuai dengan tabel 3.1 berikut.

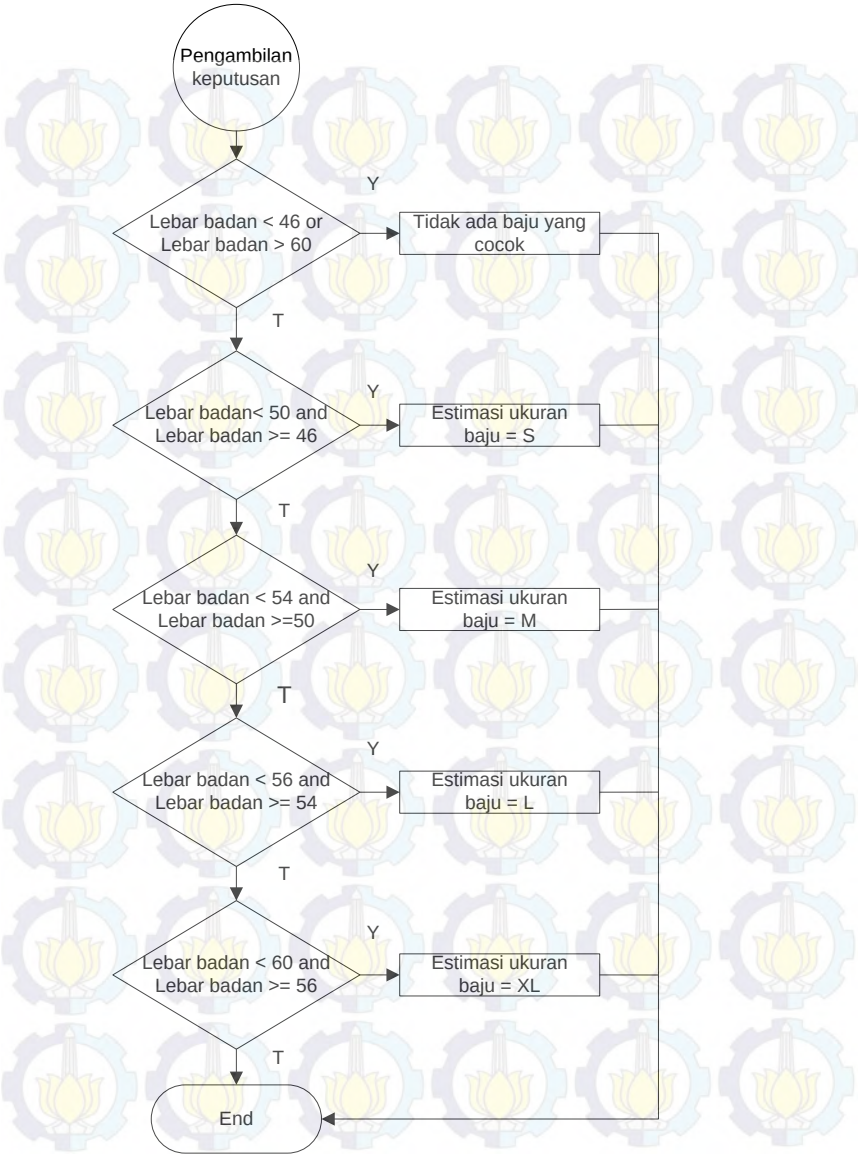
Tabel 3.1 Tabel kategori ukuran baju

NO	Kategori Baju	Lebar Badan Minimal (cm)	Lebar Badan Maksimal (cm)
1	S	46	50
2	M	50	54
3	L	54	56
4	XL	56	60
5	-	Kurang dari 46	Lebih dari 60

Gambar 3.14 Contoh kategori ukuran baju



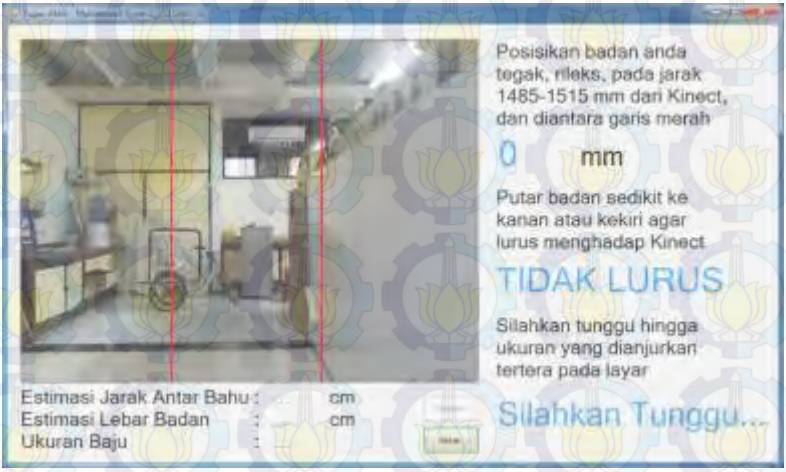
Gambar 3.15 Diagram alir proses estimasi ukuran baju



Gambar 3.16 Diagram alir proses estimasi badan

3.3.8 *Interface Program*

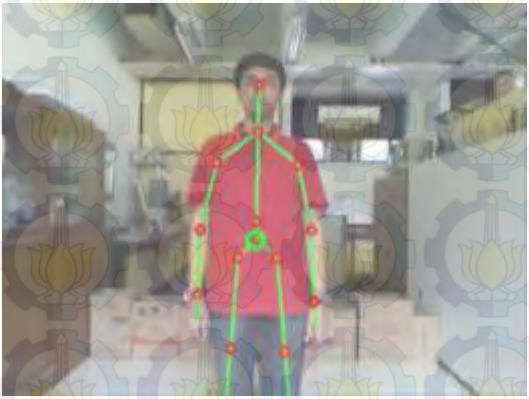
Program *interface* merupakan sebuah antarmuka antara sistem yang dirancang dengan pengguna. Melalui *interface* ini, pengguna dapat berinteraksi dengan sistem dan melihat tampilan dirinya disertai dengan *skeletal tracking*. Untuk kepentingan pengukuran maka disertakan indikator jarak, indikator lurus atau tidaknya badan terhadap kamera, estimasi jarak antar bahu, estimasi lebar badan, ukuran baju yang disarankan dan status dari sistem. Indikator jarak akan merubah gambar yang ditangkap oleh kamera Kinect menjadi merah bila pengguna berdiri diluar dari jarak yang dianjurkan (1500 mm. Selain itu sistem ini juga dilengkapi dengan NUI untuk melakukan pengukuran ulang dengan cara menempelkan tangan kanan ke dada.



Gambar 3.17 Tampilan program *interface*

Gambar yang ditampilkan pada *interface* merupakan gabungan antara citra RGB dan *skeletal tracking*. Untuk menampilkan citra RGB dan *skeletal tracking* maka *frame color stream* akan dikunci agar saat pengambilan data, data pada *color frame* tidak berubah. Lalu data *color stream* dimasukkan kedalam variabel `color`, dengan cara `cvSetData(color, pBuffer, LockedRect.Pitch)`, dimana `color` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame color stream* sedangkan `LockedRect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*. Selanjutnya `drawSkeleton(color)` dipanggil untuk proses *skeletal tracking*.

```
if (LockedRect.Pitch != 0)
{
    BYTE * pBuffer = (BYTE*) LockedRect.pBits;
    cvSetData(color, pBuffer, LockedRect.Pitch);
    drawSkeleton(color);
    pictureBox1->Image  = gcnew System::Drawing::Bitmap(
    color->width,color->height,color->widthStep,System::
    Drawing::Imaging::PixelFormat::Format32bppRgb,(System::IntPtr)
    color->imageData);
    pictureBox1->Refresh();
}
```



Gambar 3.18 Hasil perpaduan citra RGB dan *skeletal tracking*

3.4 Layar Monitor

Untuk tampilan *display* ke pengguna digunakan sebuah layar monitor ViewSonic Va1601w 15,5 inch dengan resolusi 1280x720.

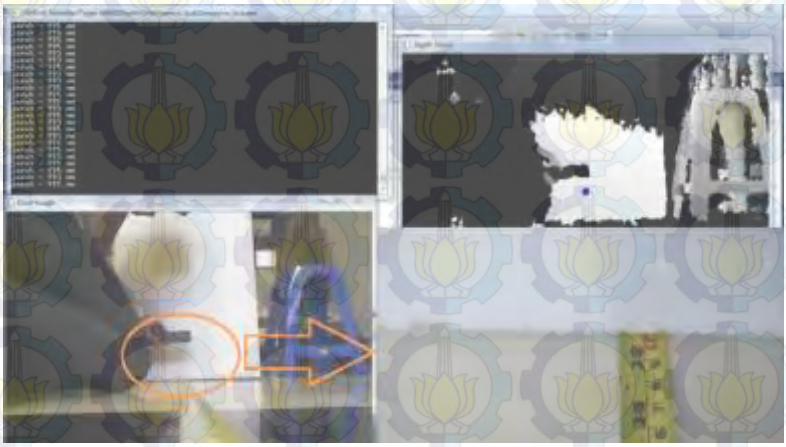
Gambar 3.19 Layar Monitor ViewSonic

BAB IV
PENGUJIAN DAN ANALISIS

Pengujian sistem dimaksudkan untuk mengetahui hasil dari sistem yang telah dirancang. Pengujian ini dimulai dengan pengukuran jarak dengan kamera Kinect. Selanjutnya pengujian akan terbagi menjadi dua yakni pada jarak optimal sistem dan pada jarak yang tidak optimal. Pengujian tersebut diantaranya yaitu pengukuran jarak antar bahu dengan metode jarak Euclidean, estimasi lebar badan dengan *Neural Networks*, dan penentuan kategori baju. Pengujian tersebut dilakukan untuk mengetahui pengaruh jarak pengguna terhadap hasil pengukuran sistem.

4.1 Pengukuran Jarak dengan Sensor Kinect

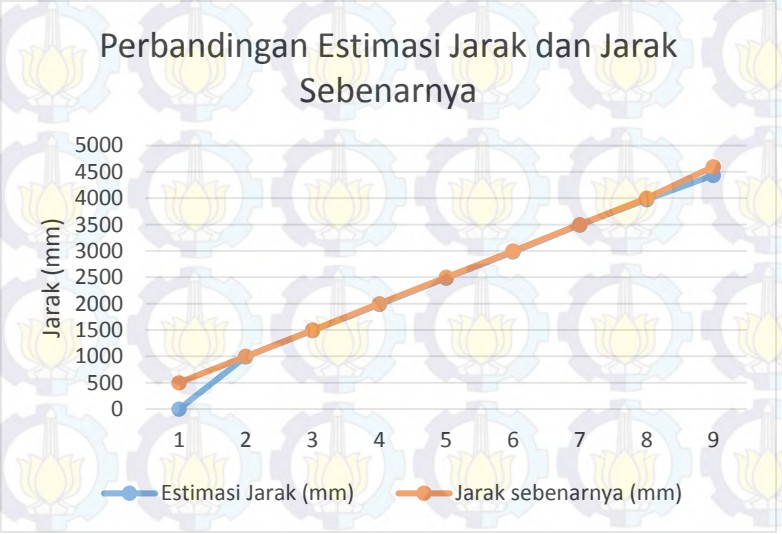
Dengan menggunakan informasi dari *Kinect IR Sensors* maka jarak antara suatu obyek dengan Kinect dapat diestimasi. Jarak tersebut tersebut umumnya disebut dengan *depth.* Data *depth* yang didapatkan dari sensor Kinect mempunyai satuan milimeter. Pengujian jarak ini dilakukan dengan cara membandingkan data *depth* yang terukur oleh sensor Kinect dengan jarak yang sebenarnya. Pengukuran jarak dilakukan pada jarak 500 mm hingga 4.500 mm dengan interval 500 mm. Pemilihan rentang jarak antara 500 mm hingga 4.500mm merupakan jarak optimal yang dapat destimasi oleh Kinect.



Gambar 4.1 Pengukuran jarak menggunakan data *depth* sensor Kinect

Tabel 4.1 Perbandingan estimasi jarak dan jarak sebenarnya

No	Estimasi Jarak (mm)	Jarak sebenarnya (mm)	Error Absolut (%)
1	0	500	100
2	993	1000	0,7
3	1489	1500	0,73
4	1986	2000	0,7
5	2487	2500	0,52
6	2980	3000	0,66
7	3493	3500	0,2
8	3975	4000	0,62
9	4434	4600	3,60



Gambar 4.2 Grafik perbandingan estimasi jarak dan jarak sebenarnya

Berdasarkan tabel 4.1 dapat dilihat bahwa sensor Kinect tidak dapat mendeteksi jarak 500 mm. Jarak minimal yang dapat diestimasi oleh sensor Kinect yaitu 1000 mm. Sensor Kinect tidak dapat mendeteksi jarak 500 mm sehingga memberikan estimasi jarak 0 mm. Sedangkan antara

jarak 1000 mm hingga 4000 mm perbandingan jarak antara hasil estimasi oleh sensor Kinect dengan jarak sebenarnya tidak mengalami perbedaan yang signifikan. Terlihat pada tabel 4.1 bahwa error absolut pada jarak tersebut kurang dari 1%. Namun pada jarak 4500 mm hasil estimasi jarak oleh sensor Kinect mengalami perbedaan yang cukup jauh dibandingkan jarak-jarak yang sebelumnya dan terdapat error abslut 3,6%. Hal ini dikarenakan pengukuran jarak dilakukan diluar batas pengukuran optimal yang dapat dilakukan oleh sensor Kinect.

4.2 Pelatihan Data pada Jarak 1500 Milimeter

4.2.1 Pengukuran Jarak Antar Bahu

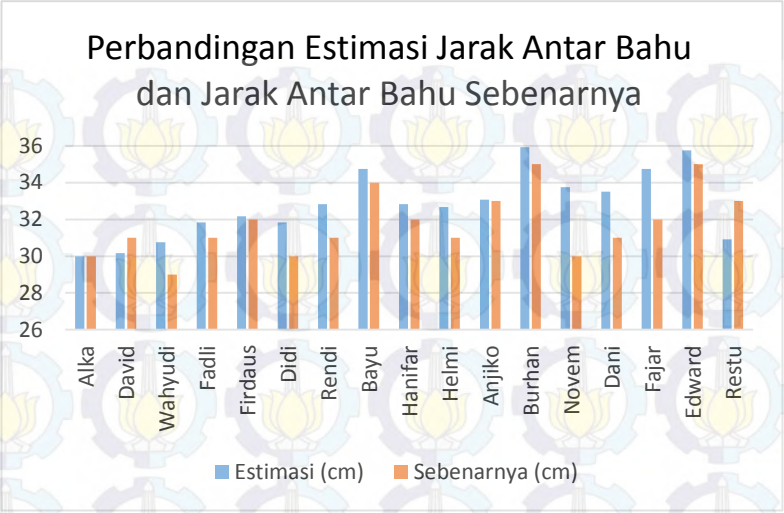
Pengujian estimasi jarak antar bahu dengan metode jarak Euclidean, yakni menentukan jarak antara dua buah titik[14]. Untuk estimasi jarak antar bahu, titik yang dihitung yakni jarak Euclidean antara sendi bahu kanan dan sendi bahu kiri yang terdeteksi oleh *skeletal tracking*. Nantinya hasil pengujian ini akan dibandingkan dengan pengukuran jarak antar bahu yang sebenarnya. Untuk pengukuran jarak antar bahu yang sebenarnya dilakukan dengan cara mengukur dengan pita meter seperti yang terdapat pada gambar 2.1. Pengujian ini dilakukan pada 17 orang dan jarak antara kamera Kinect dengan subjek pengujian yaitu sebesar 1500 mm dengan batas toleransi ± 15 mm. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 1.

Gambar 4.3 Hasil dari estimasi jarak antara bahu

Tabel 4.2 Perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Alka	30,00	30	0,00
2	David	30,17	31	2,69
3	Wahyudi	30,75	29	6,03
4	Fadli	31,83	31	2,69
5	Firdaus	32,17	32	0,52
6	Didi	31,83	30	6,11
7	Rendi	32,83	31	5,91
8	Bayu	34,75	34	2,21
9	Hanifar	32,83	32	2,60
10	Helmi	32,67	31	5,38
11	Anjiko	33,08	33	0,25
12	Burhan	35,92	35	2,62
13	Novem	33,75	30	12,50
14	Dani	33,50	31	8,06
15	Fajar	34,75	32	8,59
16	Edward	35,75	35	2,14
17	Restu	30,92	33	6,31

Pada tabel 4.2 terlihat bahwa perbandingan estimasi jarak antar bahu dengan jarak antar bahu sebenarnya terdapat error absolut yang cukup besar pada beberapa orang yaitu hingga 12,5%. Sedangkan rata-rata dari error absolut yang didapatkan yaitu 4,38% Perbedaan hasil estimasi jarak antar bahu dengan jarak antar bahu yang sebenarnya disebabkan karena penggunaan pakaian yang beragam serta pemakaian pakaian yang cenderung lebih besar dari ukuran badan yang semestinya.



Gambar 4.4 Grafik perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

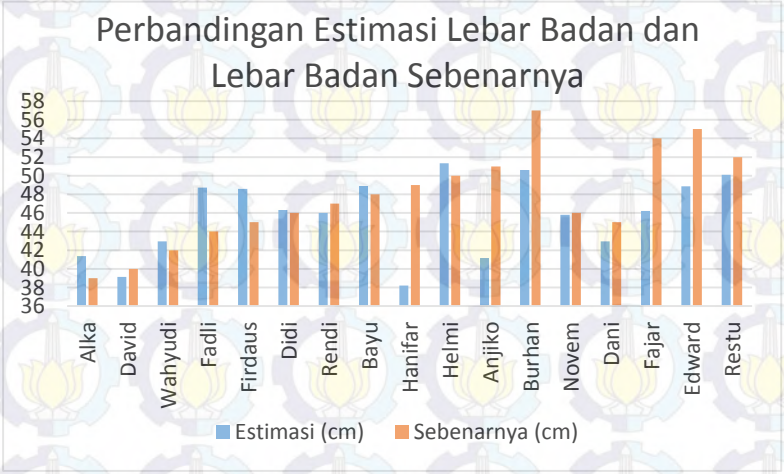
4.2.2 Estimasi Lebar Badan

Estimasi lebar badan dapat dilakukan setelah melatihkan 60 set data uji dari 12 subjek pengujian. Setiap subjek mempunyai lima set data uji. Pengambilan lima set data uji ini dimaksudkan untuk mengurangi kesalahan estimasi lebar badan akibat pergerakkan kecil seperti bernapas.

Tabel 4.3 Perbandingan estimasi lebar badan dan lebar badan sebenarnya

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Alka	41,36	39	3,10
2	David	39,14	40	1,67
3	Wahyudi	42,95	42	0,09
4	Fadli	48,72	44	3,20
5	Firdaus	48,59	45	1,51
6	Didi	46,33	46	2,86

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
7	Rendi	46,00	47	3,38
8	Bayu	48,90	48	0,04
9	Hanifar	38,20	49	7,95
10	Helmi	51,34	50	2,64
11	Anjiko	41,17	51	4,84
12	Burhan	50,62	57	2,57
13	Novem	45,81	46	2,73
14	Dani	42,94	45	1,6
15	Fajar	46,23	54	2,31
16	Edward	48,86	55	2,54
17	Restu	50,08	52	1,73



Gambar 4.5 Grafik perbandingan estimasi lebar badan dan lebar badan sebenarnya

Pengujian kali ini dilakukan pada 17 subjek. 12 subjek pertama merupakan subjek pelatihan, dan 5 lainnya merupakan subjek yang tidak

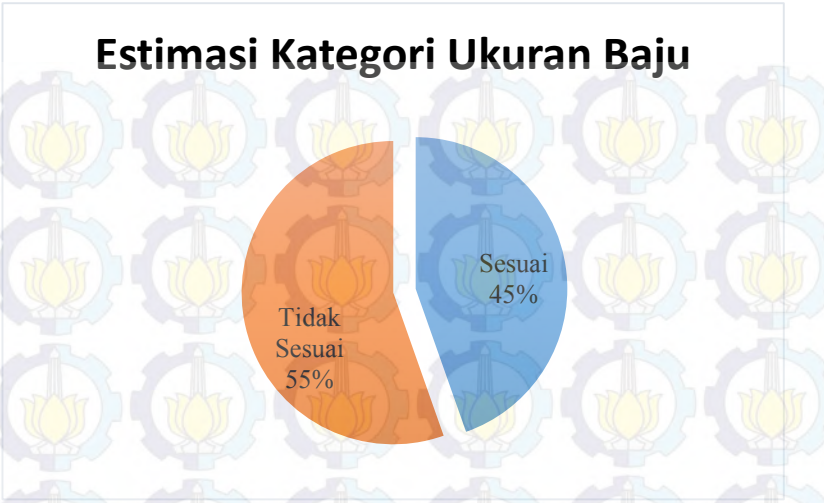
dilatihkan. Setelah melakukan pengambilan data, hasil estimasi lebar badan untuk pada 17 subjek pengujian memiliki nilai error absolut antara 0,04% hingga 7,95% dengan rata-rata nilai error absolut sebesar 2,68%. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 2.

4.2.3 Penentuan Kategori Baju

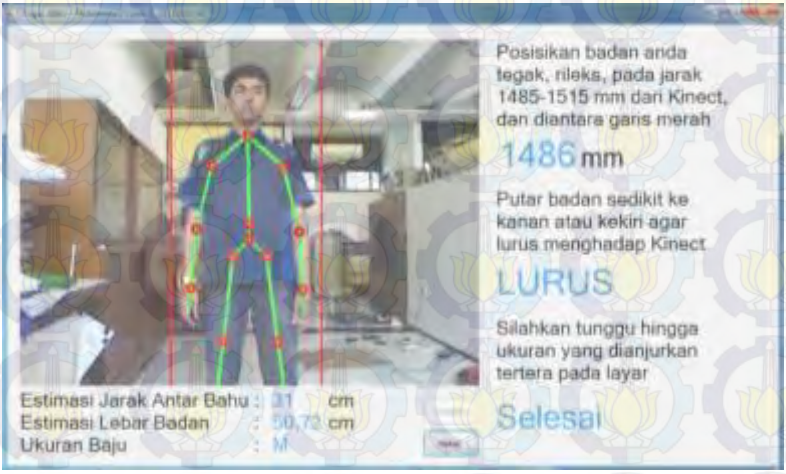
Penentuan kategori baju didapatkan dari hasil dari estimasi lebar badan yang dikonversikan kedalam kategori S, M, L, XL dan - berdasarkan pada tabel 3.1. Seperti yang terdapat pada gambar 4.7 hasil dari ukuran baju akan muncul setelah proses pengambilan data yang dilanjutkan dengan proses *feedforward* pada *Neural Networks*. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 3.

Tabel 4.4 Perbandingan estimasi ukuran baju dan ukuran sebenarnya

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Alka	-	-	Sesuai
2	David	-	-	Sesuai
3	Wahyudi	-	-	Sesuai
4	Fadli	S	-	Tidak sesuai
5	Firdaus	S	-	Tidak sesuai
6	Didi	S	S	Sesuai
7	Rendi	S	S	Sesuai
8	Bayu	S	S	Sesuai
9	Hanifar	-	S	Tidak sesuai
10	Helmi	M	M	Sesuai
11	Anjiko	-	M	Tidak sesuai
12	Burhan	M	XL	Tidak sesuai
13	Novem	-	S	Tidak sesuai
14	Dani	-	-	Sesuai
15	Fajar	S	L	Tidak sesuai
16	Edward	S	L	Tidak sesuai
17	Restu	M	M	Sesuai



Gambar 4.6 Grafik estimasi kategori ukuran baju



Gambar 4.7 Hasil dari estimasi lebar badan dan kategori ukuran pakaian

Berdasarkan hasil pengujian yang dilakukan pada 17 subjek, terdapat 3 estimasi ukuran baju yang tidak sesuai dengan ukuran baju yang sebenarnya, dikarenakan adanya pergeseran nilai estimasi lebar badan dengan lebar badan yang sebenarnya.

4.3 Pelatihan Data Pada Jarak 1200, 1500, dan 1700 Milimeter

Pengujian ini dilakukan untuk membandingkan keluaran sistem antara pelatihan data pada jarak 1500 mm dan pelatihan data pada jarak 1200 mm, 1500, serta 1800 mm. Data yang dilatihkan didapatkan dari pengambilan data pada 12 subjek. Setiap subjek akan diambil data sebanyak lima kali pada jarak 1200mm, 1500mm dan 1700mm. Sehingga setiap subjek akan memiliki 15 data untuk dilatihkan.

4.3.1 Pengukuran Jarak Antar Bahu

Terlihat pada gambar 4.8 bahwa subjek berada diluar jarak yang dianjurkan yaitu 1500 mm dan gambar yang ditankap Kinect menjadi merah sebagai indikatornya. Dapat dilihat pada tabel 4.5 estimasi jarak antar bahu pada jarak 1200 mm cukup akurat dengan rata-rata error absolut keseluruhan 6,39%. Begitu juga pada tabel 4.6 estimasi jarak antar bahu pada jarak 1500 mm dengan rata-rata error absolut keseluruhan 7,78%. Hal yang sama juga terdapat pada tabel 4.7 estimasi jarak antar bahu pada jarak 1500 mm memiliki rata-rata error absolut keseluruhan sebesar 7,46%. Untuk penentuan estimasi jarak antar bahu dapat dilakukan pada jarak 1200 hingga 1700 mm. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 4, tabel 5 dan tabel 6.

Gambar 4.8 Hasil estimasi jarak antar bahu diluar jarak yang dianjurkan

Tabel 4.5 Hasil estimasi jarak antar bahu pada jarak 1200 mm

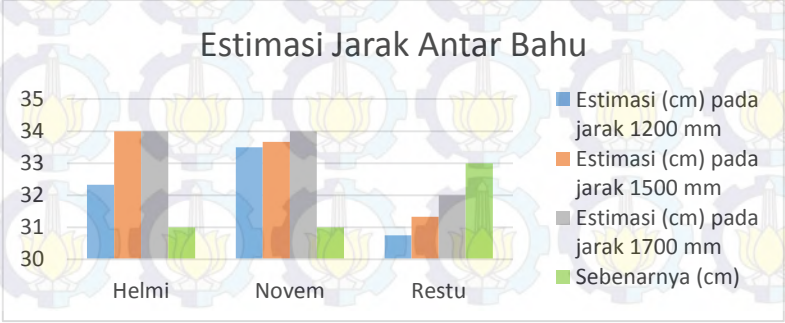
No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	32,33	31	4,30
2	Novem	33,50	31	8,06
3	Restu	30,75	33	6,82

Tabel 4.6 Hasil estimasi jarak antar bahu pada jarak 1500 mm

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	34,00	31	9,68
2	Novem	33,67	31	8,60
3	Restu	31,33	33	5,05

Tabel 4.7 Hasil estimasi jarak antar bahu pada jarak 1700 mm

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	34,00	31	9,68
2	Novem	34,00	31	9,68
3	Restu	32,00	33	3,03



Gambar 4.9 Grafik estimasi jarak antar bahu diluar jarak yang dianjurkan

4.3.2 Estimasi Lebar Badan

Pengujian kali ini juga dilakuan pada dua jarak yang berbeda yaitu pada jarak 1200 mm, 1500 mm, dan 1700 mm. Terlihat pada tabel 4.8 yaitu pada jarak 1200 mm estimasi lebar badan yang didapatkan keseluruhan rata-rata error absolut sebesar 18,85%. Pada tabel 4.9, pada jarak 1500 mm didapatkan keseluruhan error rata-rata absolut sebesar 46,36%. Sedangkan pada tabel 4.10, pada jarak 1700 mm didapatkan keseluruhan error rata-rata absolut sebesar 60,6%. Hasil pengujian pada jarak 1200 mm hingga 1700 mm secara lengkap terdapat pada lampiran tabel 7, 8, dan 9. Berdasarkan lampiran tabel 7, 8, dan 9 estimasi lebar badan pada jarak 1200 mm hingga 1700 mm cenderung tidak stabil dan tidak akurat. Hal tersebut dikarekan error pada proses pelatihan *neural networks* masih besar dan tidak bisa konvergen.

Gambar 4.10 Hasil estimasi lebar badan diluar jarak yang dianjurkan

Tabel 4.8 Hasil estimasi lebar badan pada jarak 1200 mm

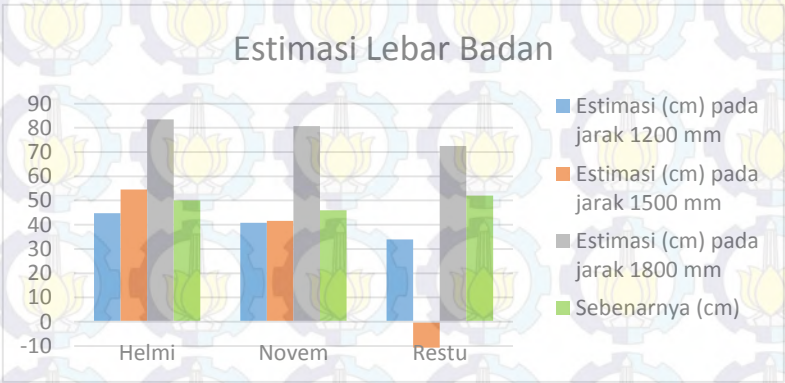
No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	44,74	50	10,52
2	Novem	40,77	46	11,38
3	Restu	33,98	52	34,66

Tabel 4.9 Hasil estimasi lebar badan pada jarak 1500 mm

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	54,51	50	9,02
2	Novem	41,62	46	9,53
3	Restu	-10,68	52	120,54

Tabel 4.10 Hasil estimasi lebar badan pada jarak 1700 mm

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	83,45	50	66,90
2	Novem	80,71	46	75,45
3	Restu	72,51	52	39,44



Gambar 4.11 Grafik estimasi lebar badan diluar jarak yang dianjurkan

4.3.3 Penentuan Kategori Baju

Berdasarkan estimasi lebar badan yang didapatkan pada tabel 4.10, tabel 4.11, dan juga tabel 4.12 maka didapatkan hasil kategori ukuran baju pada jarak 1200 mm dan juga 1800 mm adalah seperti berikut ini.

Tabel 4.11 Hasil estimasi ukuran baju pada jarak 1200 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	-	M	Tidak sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

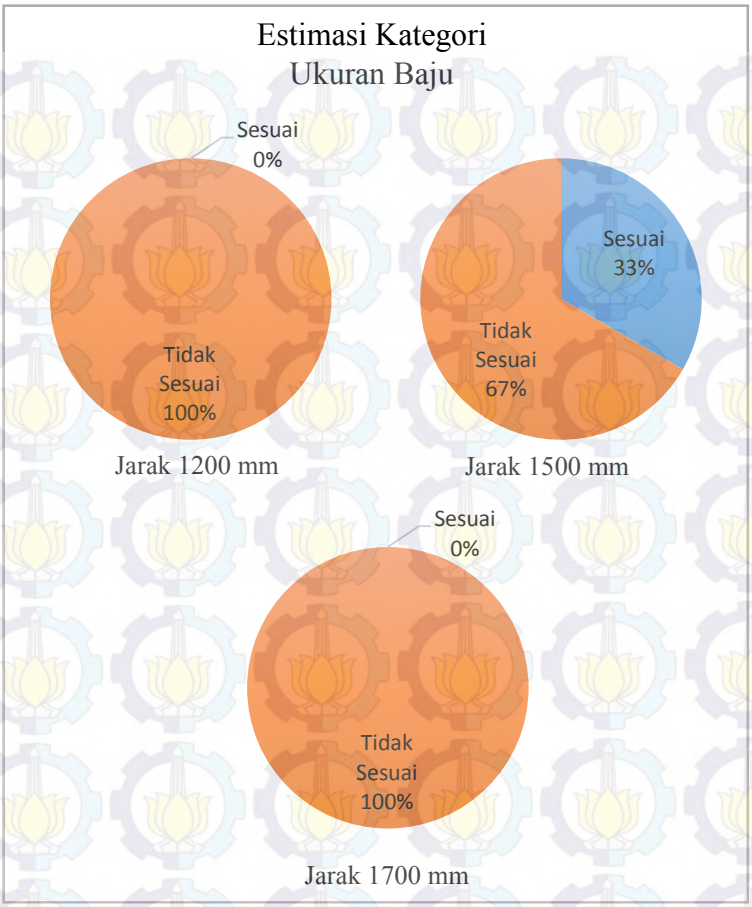
Tabel 4.12 Hasil estimasi ukuran baju pada jarak 1500 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	M	M	Sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

Tabel 4.13 Hasil estimasi ukuran baju pada jarak 1700 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	-	M	Tidak sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

Pada gambar 4.12 presentase ketidak sesuaian estimasi ukuran baju dengan ukuran baju sebenarnya mencapai 100% pada jarak 1200 mm dan 1700 mm. Sedangkan pada jarak 1500 mm presentase ketidak sesuaiannya sebesar 67%. Hal ini dikarenakan nilai estimasi lebar badan yang bergeser sangat jauh dari lebar badan sebenarnya. Hasil pengujian secara lengkap untuk proses penentuan estimasi kategori ukuran baju dapatt dilihat pada lampiran tabel 10, lampiran tabel 11, dan lampiran tabel 12.



Gambar 4.12 Grafik estimasi kategori ukuran baju diluar jarak yang dianjurkan

BAB V
KESIMPULAN DAN SARAN

Berdasarkan hasil dari uji coba sistem terhadap beberapa subjek, maka dapat ditarik beberapa kesimpulan dan saran untuk pengembangan sistem kedepan.

5.1 Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan uji coba pada sistem yaitu:

1. Pengukuran jarak yang dapat dilakukan menggunakan kamera Kinect secara optimal yaitu antara 1000mm hingga 4000mm dengan rata-rata error absolut 0,59%.
2. Hasil dari pengukuran jarak antar bahu dengan metode jarak Euclidean dapat dilakukan pada jarak antara 1200 mm hingga 1800 mm. Pada jarak 1200 mm, 1500 mm, dan 1700 mm hasil pengukuran mengalami pergeseran yang cukup kecil ditunjukkan dengan nilai rata-rata error absolut 6,39%, 7,78% dan 7,46%.
3. Pada *neural networks* yang dilatihkan pada jarak 1500 mm, sistem dapat melakukan estimasi lebar badan cukup akurat dengan nilai rata-rata keseluruhan error absolut sebesar 7,25%.
4. Pada *neural networks* yang dilatihkan pada jarak 1200 mm, 1500 mm, dan 1700 mm terdapat hasil estimasi lebar badan yang tidak akurat. Hal tersebut ditunjukkan dengan nilai rata-rata error keseluruhan estimasi lebar badan sebesar 18,85%, 46,36% dan 7,46% pada masing-masing jarak.
5. Jarak antara pengguna dengan Kinect memberikan pengaruh besar pada proses estimasi lebar badan. Pelatihan data pada jarak 1500 mm memberikan hasil yang paling baik pada sistem ini.
6. Sistem mampu memberikan saran ukuran baju berdasarkan lebar badan yang didapatkan. Tetapi hasil ukuran baju yang disarankan dapat mengalami pergeseran. Pergeseran tersebut disebabkan karena adanya pergeseran nilai estimasi lebar badan dengan lebar badan yang sebenarnya.

5.2 Saran

Untuk mengembangkan tugas akhir ini penulis memiliki beberapa saran sebagai berikut:

1. Untuk meningkatkan presisi dari estimasi lebar badan, diperlukan penambahan set data uji dengan subjek yang beragam.
2. Metode estimasi lebar badan dengan menggunakan *Neural Networks* perlu diteliti lebih lanjut.
3. Diharapkan kerja sistem tidak terganggu bila ada orang yang berdiri disekitar pengguna.
4. Melakukan pengembangan untuk sistem pengukuran secara *online*

LAMPIRAN

Tabel 1 Perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

No	Nama	Estimasi Jarak Antar Bahu (cm)												Rata- rata	Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12			
1	Alka	30	30	30	30	30	30	31	30	30	30	29	30	30,00	30	0,00
2	David	30	30	30	30	31	30	30	30	31	30	30	30	30,17	31	2,69
3	Wahyudi	31	31	31	31	30	31	31	30	30	31	31	31	30,75	29	6,03
4	Fadli	32	32	32	32	32	32	31	31	32	32	32	32	31,83	31	2,69
5	Firdaus	33	32	33	33	32	32	32	32	32	32	31	32	32,17	32	0,52
6	Didi	32	32	32	32	32	32	32	32	31	32	32	31	31,83	30	6,11
7	Rendi	33	33	33	33	32	33	33	33	32	33	33	33	32,83	31	5,91
8	Bayu	35	34	34	34	35	35	35	35	35	35	35	35	34,75	34	2,21
9	Hanifar	33	33	33	33	33	33	32	32	33	33	33	33	32,83	32	2,60
10	Helmi	33	31	33	33	33	33	33	33	33	33	33	31	32,67	31	5,38
11	Anjiko	33	33	33	33	33	33	33	33	33	34	33	33	33,08	33	0,25
12	Burhan	36	36	36	36	36	36	36	35	36	36	36	36	35,92	35	2,62
13	Novem	34	34	34	34	34	34	34	34	34	33	34	32	33,75	30	12,50
14	Dani	34	31	31	34	34	34	34	34	34	34	34	34	33,50	31	8,06
15	Fajar	35	34	35	34	35	35	34	35	35	35	35	35	34,75	32	8,59
16	Edward	36	36	36	36	35	36	36	34	36	36	36	36	35,75	35	2,14
17	Restu	31	32	31	30	30	31	31	31	31	31	31	31	30,92	33	6,31

Tabel 2 Perbandingan estimasi lebar badan dan lebar badan sebenarnya

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Alka	44,68	49,17	47,27	47,94	44	36,98	42,27	35,24	36,97	38,19	35,77	37,79	41,36	39	6,04
2	David	37,12	38,18	37,87	41,81	40,21	40,67	38	38,48	40,29	38,26	39,49	39,28	39,14	40	2,15
3	Wahyudi	41,26	47,68	42,04	39,58	39,58	37,87	39,37	46,62	47,23	43,67	44,56	45,9	42,95	42	2,25
4	Fadli	49,57	48,94	54,91	55,35	51,89	44,81	46,29	47,21	48,8	45,5	45,91	45,41	48,72	44	10,72
5	Firdaus	42,05	48,79	49,01	44,32	51,56	50,95	49,25	49,34	49,56	49,84	48,9	49,56	48,59	45	7,99
6	Didi	46,88	46,58	47,32	49,19	47,57	47,87	45,54	44,77	44,22	47,05	44,47	44,45	46,33	46	0,71
7	Rendi	47,61	47,75	48,09	47,58	46,09	48,53	48,01	48,59	46,4	47,79	46,43	46,63	46,00	47	2,13
8	Bayu	54,53	51,01	50,79	49,59	52,46	48,06	44,63	47,71	48,02	46,12	49,27	44,56	48,90	48	1,87
9	Hanifar	35,08	39,63	42,34	33,37	38,97	34,18	33,85	40,34	45,1	42,38	35,1	38,05	38,20	49	22,04
10	Helmi	41,75	50,71	58,7	54,41	55,2	55,39	51,32	47,02	52,9	49,3	47,15	52,19	51,34	50	2,67
11	Anjiko	41,4	42,55	39,9	47,05	45,61	37,35	40,07	48,53	38,69	40,14	34,46	38,28	41,17	51	19,28
12	Burhan	53,74	50,6	48,8	45,91	43,48	44,5	52,24	57,46	58,78	51,02	52,62	48,31	50,62	57	11,19
13	Novem	45,93	47,26	44,79	44	44,85	45,12	46,36	48,7	45,49	44,63	44,65	47,93	45,81	46	0,41
14	Dani	42,63	38,27	45,72	39,1	47,8	43,37	49,4	38,98	40,37	43,85	40,08	45,73	42,94	45	4,57
15	Fajar	31,28	30,56	47,01	50,23	55,25	53,2	53,56	44,41	51,57	38,14	50,67	48,82	46,23	54	14,40
16	Edward	48,94	46,97	45,54	49,77	53,6	42,15	51,6	46,92	48,02	49,96	53,117	49,75	48,86	55	11,16
17	Restu	48,75	50,06	50,01	51,42	49,49	50,33	49,99	48,9	52	49,94	51	49,05	50,08	52	3,70

Tabel 3 Perbandingan estimasi ukuran baju dan ukuran sebenarnya

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Alka	-	S	S	S	-	-	-	-	-	-	-	-	-	75,00
2	David	-	-	-	-	-	-	-	-	-	-	-	-	-	100,00
3	Wahyudi	-	S	-	-	-	-	-	S	S	-	-	-	-	75,00
4	Fadli	S	S	L	L	M	-	S	S	S	-	-	-	-	41,67
5	Firdaus	-	S	S	-	M	M	S	S	S	S	S	S	-	16,67
6	Didi	S	S	S	S	S	S	-	-	-	S	-	-	S	58,33
7	Rendi	S	S	S	S	S	S	S	S	S	S	S	S	S	100,00
8	Bayu	L	M	M	S	M	S	-	S	S	S	S	-	S	50,00
9	Hanifar	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
10	Helmi	-	M	XL	L	L	L	M	S	M	S	S	M	M	33,33
11	Anjiko	S	-	-	-	S	-	-	-	S	-	-	-	M	0,00
12	Burhan	M	M	S	-	-	-	M	XL	XL	M	M	S	XL	25,00
13	Novem	-	S	-	-	-	-	S	S	-	-	-	S	S	41,67
14	Dani	-	-	-	-	S	-	S	-	-	-	-	-	-	83,33
15	Fajar	-	-	S	M	L	M	M	-	M	-	M	S	L	8,33
16	Edward	S	S	-	S	M	-	M	S	S	S	M	S	L	0,00
17	Restu	S	M	M	M	S	M	S	S	M	S	M	S	M	50,00

Tabel 4 Hasil estimasi jarak antar bahu pada jarak 1200 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	33	33	33	31	33	33	33	31	31	33	33	31	32,33	31	4,30
2	Novem	34	34	34	33	33	34	33	34	34	33	33	33	33,50	31	8,06
3	Restu	31	31	30	31	31	31	31	31	31	30	31	30	30,75	33	6,82

Tabel 5 Hasil estimasi jarak antar bahu pada jarak 1500 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
2	Novem	34	33	34	33	34	33	34	34	34	34	33	34	33,67	31	8,60
3	Restu	31	31	31	31	31	32	32	31	32	31	31	32	31,33	33	5,05

Tabel 6 Hasil estimasi jarak antar bahu pada jarak 1700 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
2	Novem	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
3	Restu	32	32	32	32	32	32	32	32	32	32	32	32	32,00	33	3,03

Tabel 7 Hasil estimasi lebar badan pada jarak 1200 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	41,1	45.86	36.64	47.08	45,78	50,37	48,1	44.04	45.55	48,38	43,62	35,83	44,74	50	10,52
2	Novem	38,9	41,56	41,58	39,87	40,88	41,14	40,12	41,58	40,5	41,05	40,7	41,33	40,77	46	11,38
3	Restu	45,66	26,66	38,18	36,48	36,48	18,05	31,94	27,2	38,45	30,25	36,48	41,89	33,98	52	34,66

Tabel 8 Hasil estimasi lebar badan pada jarak 1500 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	51,41	48,33	56,02	58,18	55,65	56,8	56,08	57,44	57,44	58,18	48,87	49,73	54,51	50	9,02
2	Novem	38,74	38,5	37,18	37,97	46,45	40,36	39,58	51,08	41,01	39,58	48,57	40,36	41,62	46	9,53
3	Restu	21,06	-12,6	4,06	-16	-16	-3,1	-16	-3,8	-22	-23,2	-31	-8,9	-10,6	52	120,43

Tabel 9 Hasil estimasi lebar badan pada jarak 1700 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	86,29	85,75	80,21	86,29	85,75	84,65	84,65	76,88	86,71	79,14	78,77	86,29	83,45	50	66,90
2	Novem	84,94	86,54	82,33	77,44	80,01	84,55	77,23	76,05	83,42	76,81	84,35	74,82	80,71	46	75,45
3	Restu	88,86	78,63	60,98	75,19	70,91	75,19	79,36	77,96	78,63	61,74	75,19	47,48	72,51	52	39,44

Tabel 10 Hasil estimasi ukuran baju pada jarak 1200 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	S	-	M	S	-	-	S	-	-	M	8,33
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00

Tabel 11 Hasil estimasi ukuran baju pada jarak 1500 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	XL	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00

Tabel 12 Hasil estimasi ukuran baju pada jarak 1700 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	S	-	M	0,00

Listing program Visual Studio C++

```
double round(double d)
{
  return floor(d + 0.5);
}
void ambil_in(int jenis_out)
{
   DataIn[jenis_out][0]=lebar_p;
   DataIn[jenis_out][1]=lebar_s;
   DataIn[jenis_out][2]=LCdepth;
   DataIn[jenis_out][3]=RCdepth;
}
void save_trainsets(int jumlahdata, int index)
{
   ofstream outFile;
   string a ("Data (");
   string c (").txt");
   string nama_data;
   stringstream b;
   stringstream d;
   b<<jumlahdata;
   d<<index;
   nama_data= d.str()+a+b.str()+c;
   outFile.open(nama_data);
   for (int i = 0 ; i < input; i++)
   {
        outFile << DataIn[jumlahdata-1][i]<<endl;
        mbuh =jumlahdata;
   }
   for (int i = 0 ; i < Output; i++)
   {
        outFile << OutDesire[jumlahdata-1][i]<<endl;
   }
   outFile.close();
   if (jumlahdata==20)
   {
        jumlahdata++;
        stringstream b;
        b<<jumlahdata;
        nama_data=d.str()+a+b.str()+c;
        outFile.open(nama_data);
```

```
        for (int k=0; k<jumlahdata-1; k++)
        {
            for (int i = 0 ; i < input; i++)
            {
                outFile << DataIn[k][i]<<endl;
            }
            for (int i = 0 ; i < Output; i++)
            {
                outFile << OutDesire[k][i]<<endl;
            }
        }
        outFile.close();
    }
}
void ambil_out()
{
    RandomIn[0]=lebar_s;
    RandomIn[1]=LSdepth;
    RandomIn[2]=RSdepth;
    RandomIn[3]=LSdepth - chestdepth;
    RandomIn[4]=LSdepth - bellydepth;
}
void IO_txt()
{
    ifstream inFile;
    inFile.open("Input.txt");
    for (int i = 0; i<JenisOut; i++)
    {
        for (int j = 0 ; j < input; j++)
        {
        inFile >> DataIn[i][j];
        }
    }
    inFile.close();
    inFile;
    inFile.open("Output.txt");
    for (int i = 0; i<JenisOut; i++)
    {
        for (int j = 0 ; j < Output; j++)
        {
        inFile >> OutDesire[i][j];
        }
    }
```

```
    inFile.close();
}
void denormalize ()
{
    double xmin = OutDesire[0][0];
    double xmax = OutDesire[0][0];
    for(int i=0;i<JenisOut;i++)
    {
        if(xmin>OutDesire[i][0])
        {
            xmin=OutDesire[i][0];
        }
        else if(xmax<OutDesire[i][0])
        {
            xmax = OutDesire[i][0];
        }
    }
size_est = 0.5*(xo3[0]+1)*(xmax-xmin) + xmin;
}
void normalize(int pola, int jenis_out, int trainOrRandom)
{
    double xmin = DataIn[pola][pola];
    double xmax = DataIn[pola][pola];
    if(trainOrRandom==1)
    {
        for(int i=0;i<jenis_out;i++)
        {
            if(xmin>DataIn[i][pola])xmin=DataIn[i][pola];
            else if(xmax<DataIn[i][pola])
                xmax = DataIn[i][pola];
RandomIn[pola]= 2* (RandomIn[pola]-xmin) / (xmax-xmin) - 1;
    }
}

void feedforward()
{
    //----------------- input --> H1
    for (int i = 0; i < HiddenLayer1; i++)
    {
        v1[i]=bias1[i];
        for (int j = 0; j < input; j++)
        {
            v1[i] = v1[i] + (W1[i][j]*RandomIn[j]);
```

```
        }
        xo1[i] = 2 / (1+exp(-2*v1[i])) -1;
    }
    //----------------- input --> H2
    for (int i = 0; i < Output; i++)
    {
        v2[i]=bias2[i];
        for (int j = 0; j < HiddenLayer1; j++)
        {
            v2[i] = v2[i] + (W2[i][j]*xo1[j]);
        }
        xo3[i] = v2[i];
    }
}
void load_w1()
{
    ifstream inFile;
    inFile.open("w1.txt");
    for (int i = 0; i<HiddenLayer1; i++)
    {
        for (int j = 0 ; j < input; j++)
        {
        inFile >> W1[i][j];
        }
    }
    inFile.close();
}
void load_w2()
{
    ifstream inFile;
    inFile.open("w2.txt");
    for (int i = 0; i<Output; i++)
    {
        for (int j = 0 ; j < HiddenLayer1; j++)
        {
        inFile >> W2[i][j];
        }
    }
    inFile.close();
}
void load_bias1()
{
    ifstream inFile;
```

```
    inFile.open("bias1.txt");
    for (int i = 0; i<HiddenLayer1; i++)
    {
        inFile >> bias1[i];
    }
    inFile.close();
}
void load_bias2()
{
    ifstream inFile;
    inFile.open("bias2.txt");
    for (int i = 0; i<Output; i++)
    {
        inFile >> bias2[i];
    }
    inFile.close();
}
void rata2()
{
    if (kondisi <rata)
    {
        hitung[kondisi][0]=lebar_s;
        hitung[kondisi][1]=LSdepth;
        hitung[kondisi][2]=RSdepth;
        hitung[kondisi][3]=LSdepth - chestdepth;
        hitung[kondisi][4]=LSdepth - bellydepth;
    }
}
void load_train()
{
    load_w1();
    load_w2();
    load_bias1();
    load_bias2();
}
void NN_main()
{
    normalize(0,JenisOut,1);
    normalize(1,JenisOut,1);
    normalize(2,JenisOut,1);
    normalize(3,JenisOut,1);
    normalize(4,JenisOut,1);
    feedforward();
```

```
    denormalize();
}
void average()
{
    if (kondisi==rata)
    {

        for(int i=0; i<rata;i++)
        {
            RandomIn[0] = hitung[i][0];
            RandomIn[1] = hitung[i][1];
            RandomIn[2] = hitung[i][2];
            RandomIn[3] = hitung[i][3];
            RandomIn[4] = hitung[i][4];

            NN_main();
            temp[i]=size_est;
            size_avg+=temp[i];
        }
        size_avg= size_avg/rata;
        int c;
        c=size_avg*100;
        size_avg=c/100.;
        kondisi++;
    }

}
void kosongkan()
{
    for (int i=0; i<rata;i++)
    {
        hitung[i][0]=0;
        hitung[i][1]=0;
        hitung[i][2]=0;
        hitung[i][3]=0;
        hitung[i][4]=0;
        size_avg=0;
        temp[i]=0;
        kondisi=0;
        label4->Text=String::Concat("...");
        label20->Text=String::Concat("...");
        label21->Text=String::Concat("...");
    }
```

```
}
void kategori ()
{
   if ((size_avg<46) || (size_avg>60))
        {
            label4->Text= "-";
        }
   if ((size_avg<50) && (size_avg>46))
        {
            label4->Text= "S";
        }
   if ((size_avg<54) && (size_avg>50))
        {
            label4->Text= "M";
        }
   if ((size_avg<56) && (size_avg>54))
        {
            label4->Text= "L";
        }
   if ((size_avg<60) && (size_avg>56))
        {
            label4->Text= "XL";
        }
   kondisi=1000;
}
HRESULT inisialisasi_depth()
{
//====================   inisialisasi penggunaan depth
   hr= m_pNuiSensor-
>NuiInitialize(NUI_INITIALIZE_FLAG_USES_COLOR|NUI_INITIALIZ
E_FLAG_USES_DEPTH | NUI_INITIALIZE_FLAG_USES_SKELETON);
   if(SUCCEEDED(hr))
   {
//====================   kalo berhasil inisialisasi, di
create event
        m_hNextColorFrameEvent = CreateEvent(NULL, TRUE,
FALSE, NULL);

   m_hNextDepthFrameEvent=CreateEvent(NULL,TRUE,FALSE,NULL)
;
        m_hNextSkeletonEvent = CreateEvent(NULL, TRUE,
FALSE, NULL);
```

```
        hr = m_pNuiSensor-
>NuiImageStreamOpen(NUI_IMAGE_TYPE_COLOR,NUI_IMAGE_RESOLUTI
ON_640x480, 0, 2, m_hNextColorFrameEvent,
&m_pColorStreamHandle);
        if( FAILED( hr ) )return hr;
        }
//====================    buka depth stream untuk dapat
frame depth
        hr = m_pNuiSensor-
>NuiImageStreamOpen(NUI_IMAGE_TYPE_DEPTH,
NUI_IMAGE_RESOLUTION_640x480,0,2,m_hNextDepthFrameEvent,&m_
pDepthStreamHandle);
        if( FAILED( hr ) )return hr;
        hr = m_pNuiSensor-
>NuiSkeletonTrackingEnable(m_hNextSkeletonEvent,NUI_SKELETO
N_TRACKING_FLAG_ENABLE_IN_NEAR_RANGE);
        if( FAILED( hr ) )return hr;
    return hr;
}
//====================    cek sensor
HRESULT initial()
{
    iSensorCount=0;
    hr = NuiGetSensorCount(&iSensorCount);
    if(FAILED(hr))return hr;
    for (int i = 0 ; i < iSensorCount; ++i)
    {
//====================    buat index status sensor untuk di
cek statusnya, kalo gak bisa lanjut aja
        hr = NuiCreateSensorByIndex ( i, &pNuiSensor);
        if(FAILED(hr))continue;
//====================    cek statusnya, kalo OK di
inisialise
        hr = pNuiSensor-> NuiStatus();
        if(S_OK==hr)
        {
            m_pNuiSensor=pNuiSensor;
            break;
        }
//====================    kalo gak ok, release aja, gak
kepake kok
        pNuiSensor->Release();
    }
```

```
//==================    kalo ok berarti pNuiSensornya !=
NULL
   if(NULL!= m_pNuiSensor)hr= inisialisasi_depth();
//==================    kalo gak ok nuisensor 2=null
   if(NULL==m_pNuiSensor||FAILED (hr))return E_FAIL;
   else
   {
//==================    siapin frame windows penamung img
depth
      color = cvCreateImageHeader(cvSize(cColorWidth,
cColorHeight), IPL_DEPTH_8U, 4);

   depthh=cvCreateImageHeader(cvSize(cDepthWidth,cDepthHeig
ht),IPL_DEPTH_8U, cBytesPerPixel);
      color_r = cvCreateImage(cvGetSize(color), color-
>depth, 1);
      color_g = cvCreateImage(cvGetSize(color), color-
>depth, 1);
      color_b = cvCreateImage(cvGetSize(color), color-
>depth, 1);
      m_depthRGBX= new
BYTE[cDepthWidth*cDepthHeight*cBytesPerPixel];
      //cvNamedWindow("Color Image", CV_WINDOW_AUTOSIZE);
      cvNamedWindow("Depth Image",CV_WINDOW_AUTOSIZE);
   }
}
void PointSkel ( IplImage* img, Vector4 posisiskel )
{
   FLOAT coordepthX = 0, coordepthY = 0;
   NuiTransformSkeletonToDepthImage(posisiskel,
&coordepthX, &coordepthY, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX = 0, coorrgbY = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
      NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX,
(LONG)coordepthY, 0, &coorrgbX, &coorrgbY );
   cvCircle(img,cvPoint(coorrgbX,coorrgbY),10,CV_RGB(0,255,
0),5,8,0);
}
void DrawLine( IplImage* img, Vector4 joint0, Vector4
joint1)
```

```
{
   FLOAT coordepthX0 = 0, coordepthY0 = 0;
   NuiTransformSkeletonToDepthImage(joint0, &coordepthX0,
&coordepthY0, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX0 = 0, coorrgbY0 = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX0,
(LONG)coordepthY0, 0, &coorrgbX0, &coorrgbY0 );
   FLOAT coordepthX1 = 0, coorcoordepthY0 = 0;
   NuiTransformSkeletonToDepthImage(joint1, &coordepthX1,
&coorcoordepthY0, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX1 = 0, coorrgbY1 = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX1,
(LONG)coorcoordepthY0, 0, &coorrgbX1, &coorrgbY1 );
   // gambar lingkaran untuk setiap joint dan bikin
sambungan tulang
   cvLine(img, cvPoint(coorrgbX0, coorrgbY0),
cvPoint(coorrgbX1, coorrgbY1), CV_RGB(50,255,50),3,8,0);
   cvCircle(img, cvPoint(coorrgbX0, coorrgbY0),
5,CV_RGB(255,0,0),3,8,0);
}
void DrawBone(
   IplImage* img,
    const NUI_SKELETON_DATA & skel,
    NUI_SKELETON_POSITION_INDEX joint0,
    NUI_SKELETON_POSITION_INDEX joint1)
{
    NUI_SKELETON_POSITION_TRACKING_STATE joint0State =
skel.eSkeletonPositionTrackingState[joint0];
    NUI_SKELETON_POSITION_TRACKING_STATE joint1State =
skel.eSkeletonPositionTrackingState[joint1];
   //If we can't find either of these joints, exit
    if (joint0State == NUI_SKELETON_POSITION_NOT_TRACKED ||
joint1State == NUI_SKELETON_POSITION_NOT_TRACKED)
    {
        return;
    }
```

```
    const Vector4 joint0Position =
skel.SkeletonPositions[joint0];
    const Vector4 joint1Position =
skel.SkeletonPositions[joint1];
   sendi[joint0] = joint0Position;
   NuiTransformSkeletonToDepthImage(sendi[joint0],
&SenDepthX[joint0], &SenDepthY[joint0],
NUI_IMAGE_RESOLUTION_640x480);
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480,
0,(LONG)SenDepthX[joint0], (LONG)SenDepthY[joint0], 0,
&SenPosX[joint0], &SenPosY[joint0] );
    // Don't draw if both points are inferred
    if (joint0State == NUI_SKELETON_POSITION_INFERRED ||
joint1State == NUI_SKELETON_POSITION_INFERRED)
       DrawLine(img, joint0Position, joint1Position);
    // We assume all drawn bones are inferred unless BOTH
joints are tracked
    if (joint0State == NUI_SKELETON_POSITION_TRACKED &&
joint1State == NUI_SKELETON_POSITION_TRACKED)
       DrawLine(img, joint0Position, joint1Position);
}

void DrawSkeleton(IplImage* img, const NUI_SKELETON_DATA&
skel)
{
    // Render Torso
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_CENTER);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_LEFT);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_RIGHT);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SPINE);
    DrawBone(img, skel, NUI_SKELETON_POSITION_SPINE,
NUI_SKELETON_POSITION_HIP_CENTER);
```

```
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_CENTER,
NUI_SKELETON_POSITION_HIP_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_CENTER,
NUI_SKELETON_POSITION_HIP_RIGHT);
    // Left Arm
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_LEFT,
NUI_SKELETON_POSITION_ELBOW_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ELBOW_LEFT,
NUI_SKELETON_POSITION_WRIST_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_WRIST_LEFT,
NUI_SKELETON_POSITION_HAND_LEFT);
    // Right Arm
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_RIGHT,
NUI_SKELETON_POSITION_ELBOW_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ELBOW_RIGHT,
NUI_SKELETON_POSITION_WRIST_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_WRIST_RIGHT,
NUI_SKELETON_POSITION_HAND_RIGHT);
    // Left Leg
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_LEFT,
NUI_SKELETON_POSITION_KNEE_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_KNEE_LEFT,
NUI_SKELETON_POSITION_ANKLE_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ANKLE_LEFT,
NUI_SKELETON_POSITION_FOOT_LEFT);

    // Right Leg
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_RIGHT,
NUI_SKELETON_POSITION_KNEE_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_KNEE_RIGHT,
NUI_SKELETON_POSITION_ANKLE_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ANKLE_RIGHT,
NUI_SKELETON_POSITION_FOOT_RIGHT);
   LS = skel.SkeletonPositions[4];
   RS= skel.SkeletonPositions[8];
   chest = skel.SkeletonPositions[2];
   wrist= skel.SkeletonPositions[6];
   belly = skel.SkeletonPositions[1];
}

void ProcessSkel(IplImage* img)
```

```
{
       for (int i = 0; i < NUI_SKELETON_COUNT; i++)
    {
        const NUI_SKELETON_DATA & skel =
skeletonFrame.SkeletonData[i];
       switch (skel.eTrackingState)
        {
       case NUI_SKELETON_TRACKED:
           orang=1;
            DrawSkeleton(img, skel);
           cvCircle(depthh, cvPoint(SenPosX[4],
SenPosY[2]), 3,CV_RGB(0,0,255),6,0); //LS
           cvCircle(depthh, cvPoint(SenPosX[8],
SenPosY[2]), 3,CV_RGB(0,0,255),6,0); //RS
           cvCircle(depthh, cvPoint(SenPosX[2],
(SenPosY[4]+SenPosY[8])/2),3,CV_RGB(255,255,255),6,0);
//chest
           cvCircle(depthh, cvPoint(SenPosX[1],
SenPosY[1]), 3,CV_RGB(0,0,255),6,0); //belly
            break;
        case NUI_SKELETON_POSITION_ONLY:
            PointSkel(img, skel.Position);
            break;
       default:
           orang++;
        }
    }
}
void get_collor()
{
   if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextColorFrameEvent, 0))
   {
       hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pColorStreamHandle, 0,
&imageFrame);
       if (FAILED(hr))  return;
       INuiFrameTexture * pTexture =
imageFrame.pFrameTexture;
       NUI_LOCKED_RECT LockedRect;
       // Lock the frame data so the Kinect knows not to
modify it while we're reading it
       pTexture->LockRect(0, &LockedRect, NULL, 0);
```

```
        // Make sure we've received valid data
        if (LockedRect.Pitch != 0)
        {
            BYTE * pBuffer = (BYTE*) LockedRect.pBits;
            cvSetData(color, pBuffer, LockedRect.Pitch);
            ProcessSkel(color);
   cvLine(color,cvPoint(210,0),cvPoint(210,480),CV_RGB(255,
0,0),2,0);
   cvLine(color,cvPoint(420,0),cvPoint(420,480),CV_RGB(255,
0,0),2,0);
            if (merah==1)
            {
                cvSplit(color,color_b, color_g, color_r,
NULL);
                cvSet(color_r, cvScalar(255), NULL);
            cvMerge(color_b,color_g,color_r,NULL,color);
            }
            pictureBox1->Image  = gcnew
System::Drawing::Bitmap(color->width,color->height,color-
>widthStep,System::Drawing::Imaging::PixelFormat::Format32b
ppRgb,(System::IntPtr) color->imageData);
            pictureBox1->Refresh();
        }
        // We're done with the texture so unlock it
        pTexture->UnlockRect(0);
        // Release the frame
        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pColorStreamHandle,
&imageFrame);
        cvWaitKey(1);
   }
}
void get_depth()
{
   int LSPos = SenPosX[4]+ SenPosY[2]*cDepthWidth;
   int RSPos = SenPosX[8]+ SenPosY[2]*cDepthWidth;
   int LCPos = SenPosX[4]+ SenPosY[4]*cDepthWidth;
   int RCPos = SenPosX[8]+ SenPosY[8]*cDepthWidth;
   int BellyPos = SenPosX[1]+ SenPosY[1]*cDepthWidth;
   int ChestPos = SenPosX[2]+
(SenPosY[4]+SenPosY[8])/2*cDepthWidth;
   int LSpx =0;
   int RSpx =0;
```

```
    int LCpx =0;
    int RCpx =0;
    int Chestpx =0;
    int Bellypx =0;
    if
(WAIT_OBJECT_0==WaitForSingleObject(m_hNextDepthFrameEvent,
0))
    {
        hr=m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pDepthStreamHandle,0,
&imageFrame);
        if(FAILED(hr))return;
        BOOL modenear = false;
        INuiFrameTexture* tekstur;
//====================   ambil informasi depth
        hr= m_pNuiSensor-
>NuiImageFrameGetDepthImagePixelFrameTexture(m_pDepthStream
Handle,&imageFrame,&modenear,&tekstur);
        if(FAILED(hr)) goto ReleaseFrame;
        NUI_LOCKED_RECT kuncirect;
//====================   kunci frame data saat pembacaan
        tekstur->LockRect(0,&kuncirect,NULL,0);
        if (kuncirect.Pitch!=0)
//====================   dapatin nilai min-max depth yang
reliable di current frame
        {
            int minDepth = (modenear ?
NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MINIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
            int maxDepth = (modenear ?
NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MAXIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
            BYTE* runwarna = m_depthRGBX;
            const NUI_DEPTH_IMAGE_PIXEL* BuffRC;
            const NUI_DEPTH_IMAGE_PIXEL* BuffLC;
            const NUI_DEPTH_IMAGE_PIXEL* BuffRS;
            const NUI_DEPTH_IMAGE_PIXEL* BuffLS;
            const NUI_DEPTH_IMAGE_PIXEL* BuffBelly;
            const NUI_DEPTH_IMAGE_PIXEL* BuffChest;
            const NUI_DEPTH_IMAGE_PIXEL* RunBuffer =
reinterpret_cast <const
NUI_DEPTH_IMAGE_PIXEL*>(kuncirect.pBits);
```

```
//====================    end pixelnya yaitu    start +
widht*height - 1
            const NUI_DEPTH_IMAGE_PIXEL* EndBuffer =
RunBuffer + (cDepthWidth*cDepthHeight) -1;
//====================    hitung distance depth (selisihnya)
            int selisih = 0;
            selisih = EndBuffer-RunBuffer;
            while (RunBuffer<EndBuffer)
            {
                USHORT depth= RunBuffer->depth;
                BYTE intensitasnya =
static_cast<byte>(depth >= minDepth && depth <= maxDepth ?
depth % 255:0);
                *(runwarna++) = intensitasnya;
                *(runwarna++) = intensitasnya;
                *(runwarna++) = intensitasnya;
                ++runwarna;
                ++RunBuffer;
            }
            RunBuffer = RunBuffer-selisih;

            int LS_s= LS.z*1000;
            int RS_s= RS.z*1000;
            int selisih_s = abs(LS_s - RS_s);
            if ( (LS_s >= 1000) && (selisih_s < 50))
            {
                FLOAT diff_x = LS.x - RS.x;
                FLOAT diff_y = LS.y - RS.y;
                FLOAT diff_z = LS.z - RS.z;
                FLOAT tinggi_x = chest.x - wrist.x;
                FLOAT tinggi_y = chest.y - wrist.y;
                FLOAT tinggi_z = chest.z - wrist.z;
                RS_s = RS.z;
                LS_s = LS.z;
                chest_s = chest.z;
                belly_s = belly.z;
                BuffLC = RunBuffer;
                BuffRC = RunBuffer;
                BuffChest = RunBuffer;
                BuffBelly = RunBuffer;
                BuffLS = RunBuffer;
                BuffRS = RunBuffer;
                LCpx = LCPos;
```

```
                RCpx = RCPos;
                Bellypx = BellyPos;
                Chestpx = ChestPos;
                LSpx = LSPos;
                RSpx = RSPos;


                BuffChest = BuffChest + Chestpx;
                BuffBelly = BuffBelly + Bellypx;
                BuffLS = BuffLS + LSpx;
                BuffRS = BuffRS + RSpx;
                BuffLC = BuffLC + LCpx;
                BuffRC = BuffRC + RCpx;
                bellydepth = BuffBelly->depth;
                chestdepth = BuffChest->depth;
                LSdepth = BuffLS->depth;
                RSdepth = BuffRS->depth;
                LCdepth = BuffLC->depth;
                RCdepth = BuffRC->depth;
                lebar_s =
round(sqrt(pow(diff_x,2)+pow(diff_y,2)+pow(diff_z,2))*100);
            }
            lebar_p = SenPosX[8] - SenPosX[4];
            USHORT * pBuff = (USHORT*)m_depthRGBX;
            cvSetData(depthh,pBuff,kuncirect.Pitch);
            if (orang <=50)
            {
   cvCircle(depthh,cvPoint(SenPosX[4],SenPosY[2]),3,CV_RGB(
0,0,255),6,0); //LS
   cvCircle(depthh,cvPoint(SenPosX[8],SenPosY[2]),3,CV_RGB(
0,0,255),6,0); //RS
   cvCircle(depthh,cvPoint(SenPosX[2],(SenPosY[4]+SenPosY[8
])/2),3,CV_RGB(0,0,255),6,0); //chest
   cvCircle(depthh,cvPoint(SenPosX[1],SenPosY[1]),3,CV_RGB(
0,0,255),6,0); //belly
            }
            if (orang >50)
            {
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//LS
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//RS
```

```
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//chest

   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//belly
            }
        }
        cvShowImage("Depth Image",depthh);
        tekstur->UnlockRect(0);
        tekstur->Release();
        ReleaseFrame:
        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pDepthStreamHandle,&imageFram
e);
        cvWaitKey(1);
    }
}

   private: wchar_t* string_to_wchartptr(System::String^
in){
                pin_ptr<const wchar_t> wch
=PtrToStringChars(in);
                size_t sizeInBytes = ((in->Length + 1) *
2);
                wchar_t* out =
(wchar_t*)calloc(sizeInBytes, sizeof(wchar_t));
                wcscpy_s(out, sizeInBytes, wch);
                return out;
            }
   private: bool KillProcessByName(wchar_t*
szProcessToKill){

                HANDLE hProcessSnap;
                HANDLE hProcess;
                PROCESSENTRY32 pe32;
                hProcessSnap =
CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0 );
                if(hProcessSnap == INVALID_HANDLE_VALUE){
                   return false;
                }

                pe32.dwSize = sizeof(PROCESSENTRY32);
                if(!Process32First(hProcessSnap, &pe32)){
```

```
                    CloseHandle(hProcessSnap);
                    return false;
                }
                do{
                    if(!wcscmp(pe32.szExeFile,
szProcessToKill)){
                        hProcess =
OpenProcess(PROCESS_TERMINATE, 0, pe32.th32ProcessID);
                        TerminateProcess(hProcess, 0);;
                        CloseHandle(hProcess);
                    }
                }while(Process32Next(hProcessSnap,
&pe32));
                CloseHandle(hProcessSnap);
                return true;
            }
    private: System::Void button1_Click(System::Object^
sender, System::EventArgs^  e) {
    mulai=1;
    button1->Enabled = false;
    int pos_bahu;
    LONG angle;
    IO_txt();
    load_train();
    // inisialisasi sensor kinect
    hr = initial();
    //NuiCameraElevationGetAngle(&angle);
    //if (angle!=0)
    //NuiCameraElevationSetAngle(0);
    while(true)
    {
        m_pNuiSensor->NuiSkeletonGetNextFrame( 0,
&skeletonFrame );
        get_collor();
        get_depth();
        jalan=1;
        jarak = (LSdepth+RSdepth)/2;
        label1->Text = String::Concat("",(jarak));
        pos_bahu= (abs (LSdepth-RSdepth));
        if (orang>= 700) orang=100;

        if(pos_bahu <=20 && (orang<=50))
            {
```

```
                lurus=1;
                label2->Text = String::Concat("LURUS");
            }
        else lurus =0;
        if (lurus ==0) label2->Text = String::Concat("TIDAK
LURUS");
//------------ambil data-----------------//
        if (orang <=50) //ini pengganti orang=1
        {merah=1;
            if((jarak>=1480) && (jarak<=1520))
            {merah=0;
                if (lurus==1)
                {
                    rata2();
                    label20->Text= String::Concat(lebar_s);
                }}}

        if(orang>50)kosongkan();
        if(SenPosY[10]<SenPosY[1]-20) kosongkan();
        if(size_avg==0) label9->Text =
String::Concat("Silahkan Tunggu...");
        if(size_avg!=0) label9->Text =
String::Concat("Selesai");
        if (kondisi ==rata)
        {
            average();
            kondisi=0;
            RandomIn[0]=lebar_s;
            RandomIn[1]=LSdepth;
            RandomIn[2]=RSdepth;
            RandomIn[3]=LSdepth - chestdepth;
            RandomIn[4]=LSdepth - bellydepth;
            NN_main();
            label21->Text =
String::Concat("",(size_avg));//12.ToString();
            kondisi=0;
            jalan=0;
            kategori();
           }
    }
    cvReleaseImage(&color);
    cvReleaseImage(&depthh);
    cvDestroyWindow("Color Image");
```

```
   cvDestroyWindow("Depth Image");
            }
private: System::Void timer1_Tick(System::Object^  sender,
System::EventArgs^  e) {
         }
private: System::Void button5_Click(System::Object^
sender, System::EventArgs^  e) {
            RandomIn[0]=lebar_s;
            RandomIn[1]=LSdepth;
            RandomIn[2]=RSdepth;
            RandomIn[3]=LSdepth - chestdepth;
            RandomIn[4]=LSdepth - bellydepth;
            NN_main();
MessageBox::Show(size_est.ToString());
         }

private: System::Void button3_Click_1(System::Object^
sender, System::EventArgs^  e) {
         }
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
            CompTime = GetTickCount();
         }
```

---Halaman ini sengaja dikosongkan---

FINAL PROJECT – TE 141599

DESIGN AND REALIZATION OF MEN'S CLOTHES SIZE MEASURING SYSTEM BASED ON KINECT

Muhammad Soleh Gangsarestu
NRP 2211100156

Supervisor
Dr. Ir. Djoko Purwanto, M.Eng.
Ronny Mardiyanto, ST., MT., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institute Technology of Sepuluh Nopember
Surabaya 2015

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Muhammad Soleh Gangsarestu
2211100156

Dosen Pembimbing 1 : Dr. Ir. Djoko Purwanto, M.Eng
Dosen Pembimbing 2 : Ronny Mardiyanto, ST., MT.,Ph.D

ABSTRAK

Computer Vision, bidang yang melibatkan metode proses, akuisisi, analisa, dan pengolahan citra dari dunia nyata untuk menghasilkan informasi numerik maupun simbol. Informasi itu dapat digunakan untuk melakukan suatu interpretasi satuan panjang. Interpretasi tersebut dapat dimanfaatkan pada toko baju untuk sebuah sistem pengukuran badan seseorang. Sistem ini dikhususkan untuk mengukur badan pria karena lebih mudah daripada mengukur badan wanita. Penggunaannya dapat mengurangi kerugian baju yang longgar akibat dicoba oleh pelanggan, serta pelanggan merasa puas dan nyaman karena proses pengukuran yang cepat dan mendapatkan ukuran yang sesuai.

Pada Tugas Akhir dilakukan rancang bangun sistem pengukuran badan pria untuk menentukan ukuran baju. Sistem ini akan dilakukan pada suatu ruangan yang dilengkapi dengan sebuah kamera kinect, layar monitor, dan unit proses untuk interaksi pengguna dengan sistem. Kinect camera digunakan karena dapat mengambil citra RGB dan dilengkapi dengan IR *projector* serta IR *sensor* yang dapat memberikan informasi *depth*. Setelah pengguna memulai sistem, kamera kinect akan mengirimkan informasi gambar serta *depth* ke unit proses untuk melakukan pengolahan informasi visual. *Neural networks* digunakan untuk estimasi lebar badan dari pengguna, selanjutnya sistem akan memberikan saran ukuran baju S, M, L, XL, dan - (tidak ada ukuran baju yang sesuai). Pada layar monitor akan memperlihatkan data hasil pengukuran serta kategori ukuran baju yang dianjurkan untuk pengguna. Berdasarkan pengujian yang dilakukan estimasi lebar badan memilki nilai error absolut maksimal sebesar 7,95% pada jarak optimal (1500 mm).

Kata Kunci : Baju, *Computer Vision*, Kamera Kinect

DESIGN AND REALIZATION OF MEN'S CLOTHES SIZE MEASURING SYSTEM BASED ON KINECT

Muhammad Soleh Gangsarestu
2211100156

Supervisor : Dr. Ir. Djoko Purwanto, M.Eng
Co-Supervisor : Ronny Mardiyanto, ST.,MT.,Ph.D

ABSTRACT

Computer Vision is a field of study which containing some method such as acquisition, analyzing and processing of image. From those method it gives numerical and symbolic information. Those information can be used for interpret length unit. It can be used in clothing store for measuring customer's body. This system is specified for measuring men's body because it is easier than measuring women's body. This system can reduce the clothing store loss by reducing the ammount of clothes being loosen up because of the customers trials and customer will be satisfied because of the fast measuring process and have the appropriate size to the customer's body.

The Final Project designs a system which measure mens's body to determine clothes size. This body measurement system will be done in a room that is equpipped with kinect camera, monitor display, and a processing unit for the human machine interaction. Kinect camera is used because it is not only cappable of taking RGB image but also equipped with IR projector and sensor that gives depth information. After the user start the measurement system, the kinect camera will send the visual information with depth to processing unit to perform the processing of visual ingormation. Neural networks are used to estimate user's body width and then sytem will suggest shirt size category S, M, L, XL and - (there is no suitable shirt) for the user. At the monitor display will show the measurement data result and at what size category will the shirt be suggested for the user. Based on test that have been conducted the estimate chest width have maximum absolute error of 7,95% within system's optimal range (1500mm).

***Keywords**: Computer Vision, Kinect Camera, Shirt*

KATA PENGANTAR

Alhamdulillah, Puji syukur kepada Allah SWT atas kasih dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

RANCANG BANGUN SISTEM PENGUKURAN BADAN PRIA UNTUK MENENTUKAN UKURAN BAJU BERBASIS KAMERA KINECT

Dalam mengerjakan Tugas Akhir ini tentunya penulis mendapatkan banyak bantuan berupa saran, dorongan semangat, dan doa. Atas semua hal itu penulis mengucapkan banyak terimakasih dan semoga Allah akan membalas dengan sebaik-baiknya balasan. Dan sekali lagi saya ucapkan terima kasih kepada:

1. Bapak, Ibu, kakak-kakak serta seluruh keluarga yang telah memberikan dukungan semangat, motivasi, saran dan doa.
2. Bapak Dr. Ir. Djoko Purwanto, M. Eng., selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, yang diberikan selama pengerjan penelitian tugas akhir ini.
3. Bapak Ronny Mardiyanto, ST., MT., Ph. D., selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan,
4. Mas Anhar, yang telah membantu memberikan ide, serta saran-saran dalam pengerjaan Tugas Akhir ini
5. Mas Eja, yang telah meminjamkan kamera Kincet dan mengajari cara awal penggunaan kamera kinect selama pengerjaan Tugas Akhir ini
6. Mas Toni, yang turut memberikan saran dalam pengerjaan Tugas Akhir ini.
7. Mas Hilton, yang memberikan pengarahan pada awal pengerjaan Tugas Akhir.
8. Reza yang telah membantu saya diawal pengerjaan Tugas Akhir ini
9. Serta rekan-rekan seluruh asisten laboratorium elektronika yang telah membantu saya untuk pengambilan data dan pengujian data.

Surabaya, Juli 2015

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PUSTAKA

[1] N. Dao, T. Deng, and J. Cai, “Fast and automatic body circular measurement based on a single kinect,” in Asia-Pacific Signal and Information Processing Association, IEEE Annual Summit and Conference (APSIPA), 2014, pp.1-4.

[2] P. Roy, J. Nancy Staples, and J. Steve Davis, “Automatic measurement extraction for apparel from a three-dimensional body scan," in Optics and Lasers in Engineering, Elsevier Science Limited Vol. 28, 1997, pp.157-172.

[3] B. Lv, and Q. Sun Shou “Automatic measurement of scanned human body in fixed posture” in Computer-Aided Industrial Design & Conceptual Design (CAIDCD), IEEE 11th International Conference Vol. 1, 2010, pp. 575-578.

[4] Dorland, W. A. Newman, “Dorland’s Illustrated Medical Dictionary 32nd Edition,” Elsevier Health Sciences, 2011 pp.99

[5] National Aeronautics and Space Administration (NASA), “Anthropometry and Biomechanics”. <URL: http://msis.jsc.nasa.gov/sections/section03.htm> Maret 2015.

[6] Microsoft Developer Network Library. “Kinect for Windows Sensor Components and Specifications”. <URL: http://msdn.microsoft.com/en-us/library/jj131033.aspx>Maret, 2015.

[7] Z. Zhengyou, “Microsoft Kinect Sensor and Its Effect,” MultiMedia, IEEE, Vol. 19 No.2 2012, pp.4-10.

[8] H. Jungong, S. Ling, X. Dong, and J. Shotton, "Enhanced Computer Vision with Microsoft Kinect Sensor,” in Cybernetics, IEEE Transactions Vol. 43, No. 5 2013, pp. 1318-1334.

[9] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, “Depth Mapping Using Projected Patterns,” in U.S. Patent, No 8, 2008.

[10] K. Khoshelham, “Accuracy Analysis of Kinect Depth Data,” in ISPRS Workshop Laser Scanning, Vol. 38 No. 5, 2011, pp. W12.

[11] J.Shotton, A. Fitzgibbon, M. Cook, and T. Sharp, “Real-Time Human Pose Recognition in Parts from Single Depth Images,” in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference, 2011, pp. 1297-1304.

[12] S. Haykin, “Neural Networks A Comprehensive Foundation 2nd Edition” Prentice Hall, 1994 pp. 24.

[13] M. Frederic Ham, I. Konstanic, “Principles Of Neurocomputing for science & engineering,” McGraw-Hill Higher Education, 2000. pp 26-110.
[14] M. M. Deza and E. Deza “Encyclopedia of Distances 3rd Edition,” Springer Berlin Heidelberg, 2014, pp. 100
[15] Visual Studio Developer. “Visual Studio - Microsoft Developer Tools”. <URL: https://www.visualstudio.com> Mei, 2015.
[16] OpenCV Developers Team. “About”. <URL: http://opencv.org/about.html > Mei, 2015.
[17] MathWorks Technical Supports. “MATLAB” <URL: http://www.mathworks.com/products/matlab/features-b.html> Mei 2015
[18] MathWorks Technical Supports. “Improve Neural Network Generalization and Avoid Overfitting” <URL: http://www.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html> Mei 2015

BIODATA PENULIS

Muhammad Soleh Gangsarestu, seseorang yang akrab dipanggil Restu, lahir di Surabaya pada 21 Februari 1993. Penulis merupakan anak keempat dari empat bersaudara. Selama ini penulis telah menempuh pendidikan di SD Negeri 02 Pulo Gebang Jakarta (1999-2005) SMP Negeri 172 Jakarta (2005-2008) SMA Negeri 103 Jakarta (2008-2011) dan terakhir sebagai mahasiswa S1 Teknik Elektro khususnya bidang studi elektronika di ITS Surabaya (2011-2015).

Selama masa kuliah penulis aktif berorganisasi dalam UKM Fotografi ITS sebagai wakil ketua periode 2013-2014, dan menjadi koordinator Laboratorium Elektronika Dasar B 202 periode 2014-2015.

Email:
m.soleh.gangsarestu@gmail.com

BAB I
PENDAHULUAN

1.1 Latar Belakang

Setiap toko pakaian menjual berbagai macam jenis pakaian yang terbuat dari berbagai jenis bahan. Bahan yang digunakan tergantung dari penggunaan pakaian tersebut. Bahan-bahan tersebut diantaranya katun wol, sutra, linen, spandex dan poliester. Untuk melayani pelanggan maka, toko tersebut menyediakan ruang ganti untuk mencoba pakaian yang ada. Tetapi tidak semua pakaian yang dijual dapat dicoba secara langsung oleh pelanggan. Terutama kaus yang menggunakan bahan katun.

Adapun alasan yang menyebabkan kaus berbahan katun tidak diperbolehkan untuk dicoba secara langsung karena mudah menjadi longgar setelah digunakan oleh beberapa orang yang berbeda. Hal ini tentu saja menimbulkan masalah di kedua belah pihak, yaitu pihak penjual seperti toko baju dan pihak pengunjung toko. Bagi pemilik toko baju, baju yang longgar tersebut tidak layak untuk dijual, bahkan untuk dipajang juga dirasa kurang pantas. Bila hal ini terus berlanjut tentunya pihak pemilik toko akan mengalami kerugian yang cukup besar akibat dari baju yang tidak dapat dijual tersebut. Sedangkan disisi pelanggan, mereka akan merasa khawatir bila tidak bisa mencoba pakaian tersebut secara langsung, dan setelah mereka membeli dan mencoba pakaian tersebut ternyata ukuran yang mereka pilih tidak sesuai. Selain itu bila pihak toko menawarkan untuk mengukur ukuran badan pelanggan, maka pengukurannya juga masih menggunakan pita meter baju. Tidak semua pelanggan merasa nyaman dengan cara pengukuran tersebut[1]. Untuk memberikan kenyamanan yang menguntungkan kedua belah pihak baik penjual maupun pembeli, maka perlu adanya informasi tambahan mengenai ukuran dari baju tersebut[2][3].

Atas dasar kesulitan penentuan ukuran badan calon pembeli, dalam Tugas Akhir akan dilakukan rancang bangun sistem pengukuran badan pria untuk menentukan ukuran baju. Sasaran dari sistem pengukuran ini yaitu calon pembeli pria yang tidak mengetahui ukuran badannya. Mereka dapat masuk ke suatu ruangan yang dilengkapi dengan sebuah kamera kinect dan sebuah layar monitor. Untuk mendapatkan ukuran badan, pengguna berdiri didepan kamera kinect pada jarak 1.5 meter. Selanjutnya

informasi data citra akan diambil untuk keperluan pengolahan citra oleh sebuah unit proses. Dengan memanfaatkan informasi *depth* yang diambil oleh kamera kinect maka lebar badan pengguna dapat diperkirakan. Dan pada akhirnya hasil pengukuran serta kategori ukuran baju S, M, L, XL dan - (tidak ada ukuran baju yang sesuai) akan ditampilkan pada layar monitor.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimana melakukan estimasi lebar badan menggunakan informasi visual yang didapat dari kamera Kinect.
2. Bagaimana cara memberikan kategori ukuran baju berdasarkan estimasi pengukuran yang didapat.
3. Bagaimana mengkomunikasikan hasil pengukuran yang diperoleh dari pemrosesan citra kepada pengguna dengan baik.

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Mampu melakukan estimasi lebar badan menggunakan kamera Kinect.
2. Mampu memberikan kategori ukuran baju berdasarkan estimasi pengukuran yang didapat.
3. Mampu mengkomunikasikan hasil pengukuran yang diperoleh dari pemrosesan citra kepada pengguna dengan baik.

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Jenis baju yang direkomendasikan yaitu berjenis kaus atau kaus berkerah.
2. Tinggi kamera Kinect terhadap lantai ± 1 meter
3. Jarak yang dibutuhkan untuk pengukuran ±1.5 meter.
4. Posisi badan tegak lurus dengan kamera.
5. Posisi bahu lurus dan sejajar.
6. Pengkuran dilakukan pada posisi yang diperlihatkan oleh program *interface.*
7. Menggunakan kaus, kaus berkerah, atau kemeja saat pengukuran.
8. Tidak bergerak selama pengukuran dilakukan.
9. Tidak ada orang lain pada area pengukuran.

1.5 Metodologi Penelitian

Dalam penyelesaian Tugas Akhir ini digunakan metodologi sebagai berikut :

Gambar 1.1 Metodologi Tugas Akhir

1. **Studi Literatur**

   Pada tahap studi literatur dilakukan pengumpulan teori dan data untuk digunakan pada Tugas Akhir. Teori dan data yang digunakan diambil dari buku-buku, jurnal, proceeding, dan artikel-artikel di internet, meliputi:

   - Konfigurasi kamera kinect dengan Visual Studio 2010.
   - Penggunaan *library* OpenCV pada pemrosesan citra kamera kinect.
   - Mempelajari bagaimana kamera kinect dapat mengambil informasi *depth.*
   - Mempelajari bagaimana mengolah informasi *depth.*
   - Mempelajari metode *skeletal tracking* yang telah disediakan oleh Kinect SDK.

- Mempelajari bagaimana cara mengolah informasi dari *skeletal tracking.*
- Mempelajari dasar-dasar dari *backpropagation neural network*

2. **Perancangan Hardware**
   Pada Tugas Akhir ini meliputi *hardware* yang digunakan yaitu kamera kinect sebagai input visual serta data *depth* dan layar monitor sebagai ouput visual. Semua proses pada kamera Kinect dan monitor akan dilakukan oleh bagian *processing unit* berupa laptop.

3. **Perancangan Software**
   Untuk mennggunakan kamera Kinect maka langkah pertama yang dilakukan pada tahap perancangan *software* adalah membuat langkah-langkah program untuk menginisialisasi. Inisialisasi kamera kinect meliputi *RGB imaging*, *depth imaging*, dan *skeletal tracking*. *Skeletal tracking* digunakan untuk menentukan titik pengukuran yang akan dicari nilai *depth*-nya. Data -data *depth* ini nantinya akan digunakan sebagai *data sets* yang akan dilatihkan dengan *Backpropagation Neural Network* untuk menentukan ukuran lebar badan pria. Dan berdasarkan ukuran tersebut nantinya akan di masukkan kedalam katergori ukuran baju yang ada

4. **Pengujian Sistem**
   Pengujian Sistem dilakukan untuk memperoleh tingkat ketelitian dari sistem yang telah dirancang. Pengujian dilakukan secara bertahap yaitu pengujian untuk menentukan estimasi jarak antar bahu dan estimasi lebar badan pada subjek pelatihan dan subjek yang tidak dilatihkan pada jarak 1500 mm. Pengujian berikutnya dilakukan dengan memberikan variasi jarak yaitu, 1200mm, 1500mm dan 1700mm.

5. **Penulisan Laporan Akhir**
   Tahap penulisan laporan Tugas Akhir dilakukan pada saat tahap pengujian sistem dimulai serta setelahnya.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

- **Bab 1 : Pendahuluan**
  Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, sistematika penulisan, metodologi, dan relevansi.
- **Bab 2 : Dasar Teori**
  Bab ini menjelaskan tentang teori-teori dasar yang menunjang dalam pengerjaan tugas akhir ini. Teori yang digunakan meliputi teori dasar pengambilan citra RGB, pengambilan citra kedalaman (*depth)*, konversi data *depth*, *skeletal tracking, Backpropagation Neural Network*
- **Bab 3: Perancangan Sistem**
  Bab ini menjelaskan mengenai cara kerja sistem yang meliputi *hardware* maupun *software* untuk melakukan pengukuran badan pria dan memberikan rekomendasi ukuran pakaian berdasarkan hasil pengukuran yang telah didapatkan melalui *image processing* serta *Backprpagtion Neural Network*
- **Bab 4 : Pengujian dan Analisis**
  Bab ini menjelaskan mengenai data hasil pengujian yang disertai dengan analisa.
- **Bab 5 : Penutup**
  Bab ini menjelaskan mengenai kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran yang menunjang untuk pengembangan dimasa yang akan datang.

1.7 Relevansi

Mata kuliah yang mendukung tugas akhir ini adalah Dasar Interaksi Manusia Mesin dan Dasar Elektronika Cerdas. Serta beberapa refrensi mengenai kamera Kinect. Hasil dari Tugas Akhir ini diharapkan dapat dikembangkan lagi untuk ditingkatkan akurasi dan kehandalannya. Aplikasi dari sistem ini dapat diterapkan pada toko baju baik secara langsung maupun toko baju *online*.

---Halaman ini sengaja dikosongkan---

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

Tinjauan pustaka dalam bab ini menjelaskan mengenai sistem-sistem yang berhubungan tugas akhir ini serta telah diimplementasikan oleh penulis-penulis lain. Sedangkan dasar teori menjelaskan tentang teori yang mendukung keseluruhan sistem pada tugas akhir ini.

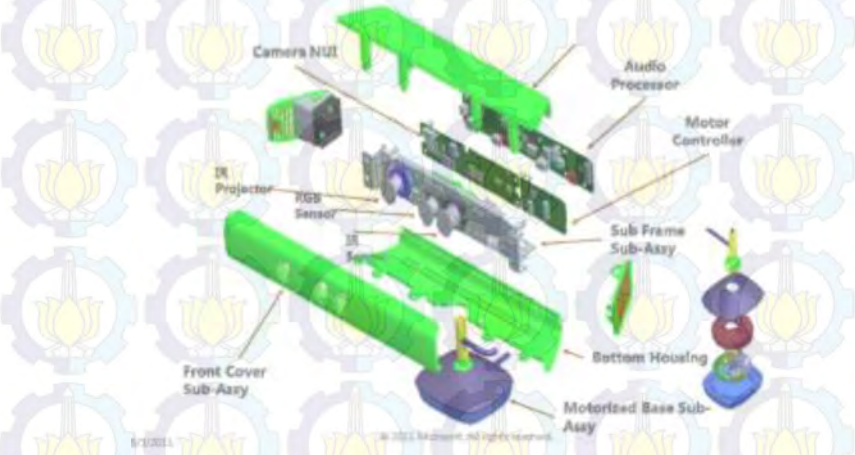
2.1 Antropometri

Antropometri dapat diartikan sebagai ilmu yang berkaitan dengan pengukuran ukuran manusia seperti berat dan proporsi badan[4]. Antropometeri memiliki peranan yang penting pada industri pakaian, ergonomis, dan arsitektur dimana data statistik mengenai distribusi dimensi tubuh pada suatu lingkungan populasi yang akan digunakan untuk mengoptimalkan suatu produk. Antropometri ini dipengaruhi oleh gaya hidup, nutrisi serta etnik dari suatu populasi, sehingga perlu adanya pembaruan secara berkala. Salah satu data antropometri yang digunakan dalam industri pakaian yaitu *chest breadth. Chest breadth* seperti yang terlihat pada gambar 2.1 merupakan lebar badan yang diukur dari ujung ketiak ke ujung ketiak lainnya[5]. Kegunaan dari *chest breadth* dalam dunia industri pakaian yaitu untuk menentukan lebar ukuran baju yang diproduksinya.

Gambar 2.1 *Pengukuran* chest breadth *oleh NASA*[5]

2.2 Kinect

Kinect merupakan seperangkat media yang digunakan oleh Microsoft untuk mendeteksi gerakan pada Xbox 360, Xbox One, dan komputer Windows. Dengan menggunakan Kinect user dapat berinteraksi secara realtime dengan perangkat-perangkat tersebut tanpa pengontrol lain. Interaksi tersebut dapat dilakukan melalui Natural User Interface (NUI)[6]. Sedangkan NUI merupakan sebuah interface yang berusaha menjadikan interaksi antara pengguna dengan suatu perangkat teknologi semudah mungkin sehingga terasa natural. Pada Kinect teradapat dua buah kamera, yaitu kamera RGB dan *infra red depth finding camera*. Selain kamera, Kinect juga dilengkapi dengan multi-array microphone.



Gambar 2.2 Bagian dari kamera Kinect[6]

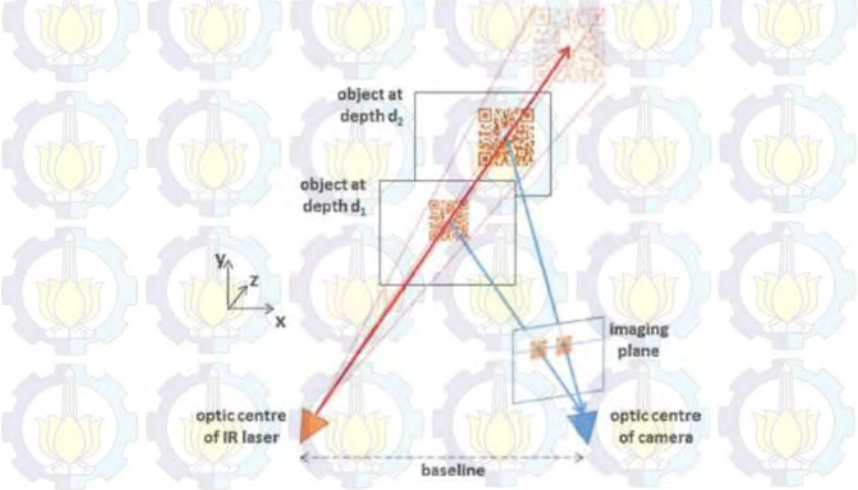
Kamera RGB pada Kinect mempunyai resolusi pengaturan dasar resolusi sebesar 640 x 480 dengan kecepatan 30 *frame* per detik dan kemampuan maksimal dari kamera RGB Kinect yaitu menghasilkan resolusi sebesar 1280 x 1024 dengan *frame* per detik yang lebih lambat. Sedangkan untuk *infrared depth finding camera* terdiri dari dua bagian utama yaitu *IR laser projector* dan *IR camera* yang dapat menghasilkan resolusi maksimal sebesar 640 x 480 dengan kecepatan 30 *frame* per detik. *IR laser projector* akan disebarkan secara acak yang pada akhir-nya akan mengenai suatu objek dan memantulkan cahaya inframerah. Pantulan

tersebut ditangkap oleh *IR camera* yang selanjutnya digunakan untuk pengolahan data kedalaman (*depth)* suatu objek pada suatu *frame*[7].

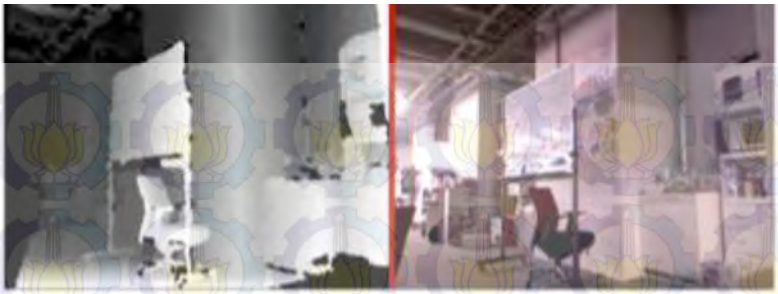
2.2.1 Citra Kedalaman *(Depth Imaging)*

Untuk memperoleh sebuah citra kedalaman maka diperlukan data kedalaman *(depth)*. Dalam kamera Kinect, data kedalaman diperoleh dari proses pemancaran dan penangkapan cahaya inframerah dari *Kinect Depth Sensors*. Posisi geometri antara pemancar inframerah (*IR* projector) dengan penangkap inframerah serta pola titik dari inframerah telah diketahui[8]. Jika titik inframerah yang dipancarkan dapat dicocokkan dengan titik yang ada pada citra yang sedang diamati maka data *depth* akan didapatkan.

Data *depth* yang didapatkan dapat ditampilkan menjadi suatu citra kedalaman *(Depth Image)*. Pada umumnya *depth image* menampilkan perubahan intensitas warna. Dimana perubahannya sesuai dengan jarak antara objek dengan kamera. Pemilihan warna yang digunakan untuk menampilkan *depth image* mulai dari putih sampai hitam, tergantung dari jarak objek terhadap Kinect. Contohya untuk jarak objek yang terlalu dekat dengan Kinect akan berwarna putih atau lebih terang, sedangkan objek yang jauh dengan akan berwarna gelap. Tidak semua objek yang tertangkap oleh Kinect mempunyai nilai



Gambar 2.3 Pengambilan data *depth*[8]



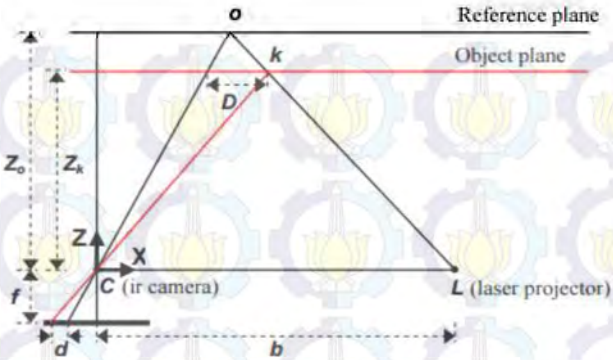
Gambar 2.4 *Depth image* dan rgb *image* pada Kinect

depth, hal ini dikarenakan *depth sensors* pada kinect mempunyai jarak minimum dan jarak maksimum untuk mendapatkan data *depth*. Karena *depths sensors* pada kinect menggunakan pantulan sinar inframerah, maka objek yang tidak dapat memantulkan sinar inframerah tidak akan memiliki nilai *depth* dan akan berwarna hitam. Hasil dari data-data *depth* tersebut selanjutnya di representasikan pada sebuah citra yang ada pada gambar 2.4

2.2.2 Depth Map

Dalam komputer grafis 3 dimensi, *depth map* adalah sebuah gambar atau *image channel* yang berisikan informasi jarak suatu permukaan suatu objek dari sebuah sudut pandang tertentu. Informasi jarak pada *depth map* umumnya dinyatakan dengan koordinat Z. Koordinat Z pada *depth map* tidak dapat disamakan dengan koordinat Z dunia, karena koordinat Z pada *depth map* bersifat relatif terhadap sudut pandang kamera.

Metode yang digunakan pada Kinect untuk menentukan estimasi jarak (*depth data*) yaitu proses tringaulasi [9]. Seperti pada gambar 2.5 yang menunjukan hubungan antara jarak dari suatu titik objek *k* ke sensor relatif terhadap bidang referensi diukur *disparity d*. Untuk mendapatkan sistem koordinat 3 dimensi, di asumsikan origin dari sistem koordinat *depth* berada pada pusat perspektif dari kamera inframerah. Dimana sumbu Z ortogonal terhadap *image plane* menuju objek. Sumbu X tegak lurus terhadap sumbu Z pada arah baseline *b* amtara kamera inframerah dan pemancar inframerah, sedangkan sumbu Y tegak lurus terhadap X dan Z dan menjadikan aturan koordinat tangan kanan.



Gambar 2.5 Representasi hubungan *depth-disparity*[10]

Koordinat *image plane* dengan *depth data* dimodelkan menjadi seperti berikut ini[10].

$$Z_k = \frac{Z_o}{1+\frac{Z_o}{fb}d} \quad (2\text{-}1)$$

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x) \quad (2\text{-}2)$$

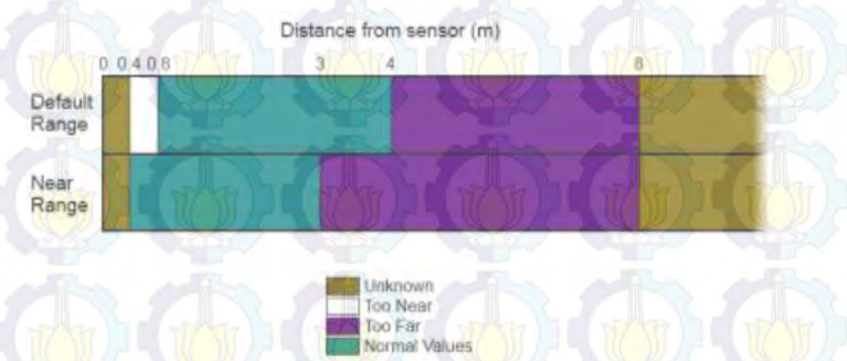
$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y) \quad (2\text{-}3)$$

Pada persamaan (2-1) Z_k merupakan penurunan model matematika untuk *depth* dari *disparity* yang diamati dengan parameter konstan jarak dari referensi pola inframerah (Z_o), lebar fokus (*f*), *base length* (*b*), dan *disparity* yang terukur (*d*). Dimana Z_o, *f*, dan *d* dapat ditentukan melalui kalibrasi. Sedangkan pada persamaan (2-2) dan (2-3) X_k dan Y_k merupakan koordinat citra dari titik *principal offsets* x_o dan y_o, sedangkan δ_x dan δ_y merupakan koefisien distorsi lensa[10].

2.2.3 Depth Space Range

Depth space range merupakan sebuah rentang jarak yang digunakan Kinect untuk mengukur *depth* dari suatu objek. Kinect mempunyai 2 jenis mode untuk melakukan pengukuran *depth*. Mode tersebut yaitu *deffault range* dan *near range*. Pada mode *deffault* maka rentang jarak pengukuran yang efektif yaitu antara 0,8 meter hingga 4

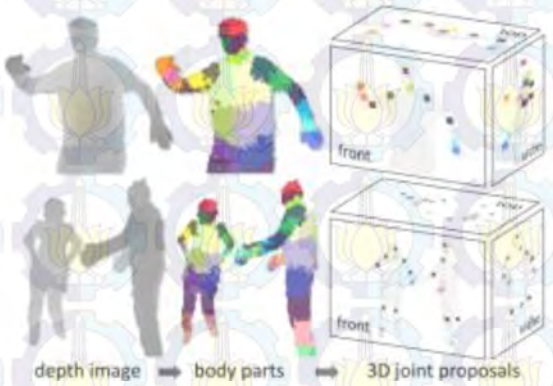
meter. Sedangkan pada mode *near* rentang jarak pengukuran yang efektif yaitu anatar 0,4 meter hingga 3 meter[6].



Gambar 2.6 Kinect *depth space range*[6]

2.2.4 Skeletal Tracking

Natural User Interface akan menjadi lebih mudah dan berjalan lebih lancar dengan bantuan *skeletal tracking*. Dalam *skeletal tracking* pada Kinect SDK 1.8, ia dapat menampilkan maksimal 2 orang dengan maksimal 20 sendi untuk setiap orangnya dan dapat mengenali 6 orang sekaligus. Sendi ini lah yang dipilih menjadi fitur untuk menentukan lebar badan yang akan di perkirakan.



Gambar 2.7 Proses dari *skeletal tracking*[11]

Skeletal tracking dihasilkan melalui dua tahapan yang utama yaitu melalui *depth image* setiap bagian badan diberikan disegmentasi. Segmentasi ini dibagikan berdasarkan posisi sendi. Segmentasi tersebut dilakukan dengan *randomized decision forest* yang akan melakukan pembelajaran dari *depth image* dengan bagian *skeletal* yang sebelumnya telah diketahui[11]. Setelah bagian tubuh telah berhasil disegmentasi langkah berikutnya yaitu memberikan estimasi dimana posisi sendi berada. Penentuan posisi sendi dilakukan dengan algoritma *mean shift* untuk mendapatkan hasil distribusi probabilitas yang kuat (*robust*).

Mode yang terdapat pada *skeletal tracking* yaitu ada 2, yang pertama yaitu *deffault mode*. Mode pertama ini dapt mengenali 20 sendi yang ada pada manusia dan hasilnya akan optimal jika penggunanya berdiri. Sedangkan yang kedua yaitu *seated mode* yang dapat melakukan *tracking* 10 sendi. Untuk jenis sendi yang ada pada masing-masing mode dapat dilihat pada tabel 2.1[6].

Tabel 2.1 Perbandingan sendi antara *default mode* dan *seated mode*

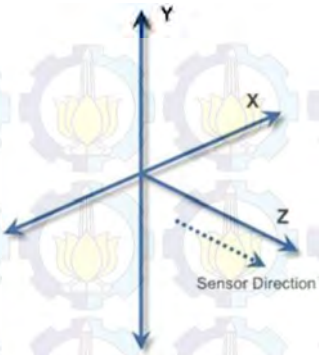
No.	Sendi	Deffault Mode	Seated Mode
1	Hip Center	Ada	Tidak
2	Spine	Ada	Tidak
3	Shoulder Center	Ada	Ada
4	Head	Ada	Ada
5	Shoulder Left	Ada	Ada
6	Elbow Left	Ada	Ada
7	Wrist Left	Ada	Ada
8	Hand Left	Ada	Ada
9	Shoulder Right	Ada	Ada
10	Elbow Right	Ada	Ada
11	Wrist Right	Ada	Ada
12	Hand Right	Ada	Ada
13	Hip Left	Ada	Tidak

No.	Sendi	Default Mode	Seated Mode
14	Knee Left	Ada	Tidak
15	Ankle Left	Ada	Tidak
16	Foot Left	Ada	Tidak
17	Hip Right	Ada	Tidak
18	Knee Right	Ada	Tidak
19	Ankle Right	Ada	Tidak
20	Foot Right	Ada	Tidak

Pada *frame* yang ada *skeleton* dapat mempunyai dua macam kondisi *tracking*. Kondisi yang pertama yakni *tracked*, pada kondisi ini *skeleton* yang ada akan memberikan informasi posisi koordinat sendi yang ada. Sedangkan kondisi yang kedua yaitu *position only* merupakan kondisi dimana *frame* akan memberikan informasi mengenai posisi dari penggunanya, namun tidak ada informasi koordinat dari sendi

2.2.5 Sekeleton Space

Ketika kondisi *tracking skeleton* berada dalam kondisi *tracked* maupun *position only* maka setiap *frame*, *depth image* akan ditangkap dan diproses oleh Kinect *runtime* menjadi *skeleton data*. *Skeleton data* berisikan posisi tiga dimensi untuk *skeleton* manusia. Pada *skeleton space* ini, posisi koordinat ditunjukkan oleh sumbu garis x,y, dan z.[6] Tetapi *depth space* dan *skeleton space* tidak bisa disamakan. Hal ini karena titik origin mereka tidaklah sama. Bila dilihat pada layar, *skeleton space* mempunyai origin (0,0,0) yang terletak tepat ditengah layar. Sedangkan *depth space* mempunyai origin (0,0,0) yang terletak pada pojok kiri atas layar. Selain itu perbedaanya sumbu z pada *skeleton space* di definisikan dengan standart satuan meter berbeda dengan *depth space* dimana sumbu z didefinisikan dengan standart satuan milimeter. Koordinat yang digunakan pada *skeleton space* menggunakan aturan tangan kanan dimana sumbu z positif memanjang seiring dengan arah sensor Kinect. Sumbu y positif memanjang ke atas dan sumbu positif x memanjang ke arah kanan seperti yang terlihat pada gambar 2.8[6].



Gambar 2.8 *Skeleton space*[6]

2.3 Artificial Neural Networks

Menurut Haykin, S *Artificial Neural Network* adalah sebuah prosesor paralel yang tersdistribusi secara masif yang secara natural mempunyai kecenderungan untuk menyimpan informasi pengalaman yang nantinya dapat digunakan lagi[12]. Hal tersebut menyerupai seperti otak dalam dua aspek yaitu:

a. Informasi pengalaman tersebut didapatkan melalui proses pembelajaran.
b. Kekuatan antara koneksi *interneuron* dikenal sebagai bobot sinaptik yang digunakan untuk menyimpan informasi pengalaman.

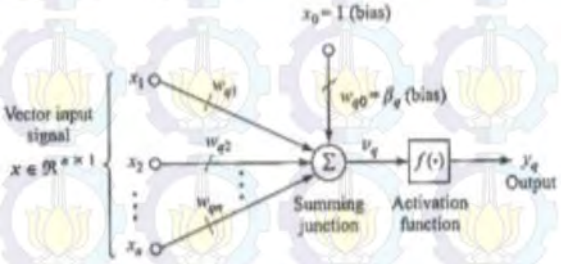
Artificial neural networks atau Jaringan Saraf Tiruan (JST) merupakan sebuah metode pembelajaran berbasis stiatistik. Metode ini terinspirasi dari jaringan saraf manusia dimana otak sebagai unit pengolah yang utama. Pada jaringan saraf manusia, setiap saraf terhubung dengan saraf lainnya melalui sinapsis sedangkan pada jaringan saraf tiruan setiap saraf diibaratkan sebuah *node* yang saling terhubung dengan *node* lainnya melalui sebuah garis yang memiliki bobot nilai. Bobot nilai dimaksudkan seperti pengalaman manusia dalam menerima rangsangan maupun pembelajaran yang diulang secara terus menerus hingga akhirnya manusia dapat mengerti

Secara sederhana jaringan saraf tiruan ini akan belajar secara terus menerus sesuai dengan tujuan yang telah ditentukan sebelumnya.

2.3.1 Struktur Neural Networks [13]

Dasar struktur dari jaringan saraf tiruan dapat dilihat pada gambar 2.9. Terdapat 3 komponen dasar pada jaringan saraf tiruan yaitu:

a. Terdapat beberapa set dari sinapsis yang saling terkait dengan *synaptic weights*, dimana komponen vektor x_j dengan j = 0, 1, 2, . . . , n. Setiap vektor x_j menjadi masukkan ke sinapsis j dan terhubung dengan neuron q melalui *synaptic weight* w_{qj}. Penulisan *subscript* yang pertama memiliki keterkaitan *neuron* tertentu, sedangkan *subscript kedua* memiliki keterkaitan dengan elemen input vektor.
b. *Device* penjumlah berfungsi untuk menjumlahkan semua sinyal yang menuju pada *device* penjumlah, dimana setiap masukkan dikali dengan *synaptic weight dan bias* (khusus x_0) yang bersangkutan lalu dijumlahkan. Semua operasi ini terjadi pada penjumlah v_q dan masih merupkan operasi linier.
c. Fungsi aktivasi $f(\cdot)$ akan membatasi amplitudo dari keluaran neuron y_q, bila menggunakan fungsi aktivasi nonlinier.



Gambar 2.9 Struktur dasar *Neural Networks*

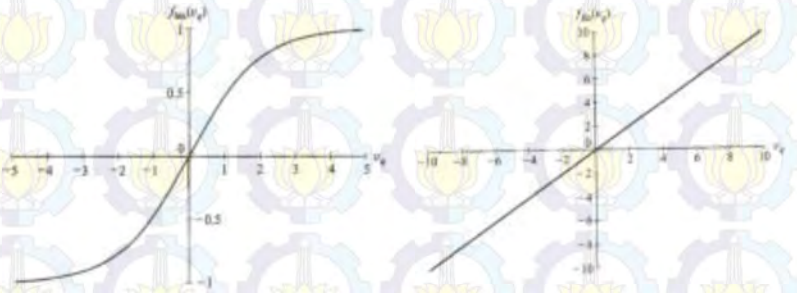
Secara matematis, operasi pada *Neural Networks* dalam gambar 2.9 dapat dituliskan menjadi persamaan berikut:

$$v_q = \sum_{j=0}^{n} w_{qj}\, x_j \qquad (2\text{-}4)$$

$$y_q = f(v_q) \qquad (2\text{-}5)$$

2.3.2 Fungsi Aktivasi [13]

Fungsi aktivasi terdiri dari berbagai macam jenis seperti fungsi *unit step*, linier, trigonometeri, hiperbolik, dan sigmoid. Pada metode *backporpagation* kombinasi dari dua atau lebih fungsi aktivasi akan menentukan penggunaan dari *Neural Networks*, contohnya kombinasi fungsi aktivasi linier dan hiperbolik tangen sigmoid (bipolar sigmoid) menghasilkan regresi dari masukan-masukan yang diberikan.



(a) (b)

Gambar 2.10 (a) Fungsi Aktivasi hiperbolik tangen sigmoid dan fungsi aktivasi linier (b)

Fungsi aktivasi bipolar sigmoid memiliki rentang keluaran antara -1 hingga -1. Sedangkan untuk fungsi aktivasi linier ia tidak memiliki rentang keluaran. Fungsi aktivasi bipolar sigmoid memiliki persamaan (2-6) dengan turunan (2-7) sedangkan fungsi aktivasi linier memiliki persamaan (2-8).

$$y_q = f_{hts}(v_q) = \frac{1 - e^{-2v_q}}{1 + e^{-2v_q}} \tag{2-6}$$

$$g_{hts}(v_q) = \left(1 + f_{hts}(v_q)\right)\left(1 + f_{hts}(v_q)\right) \tag{2-7}$$

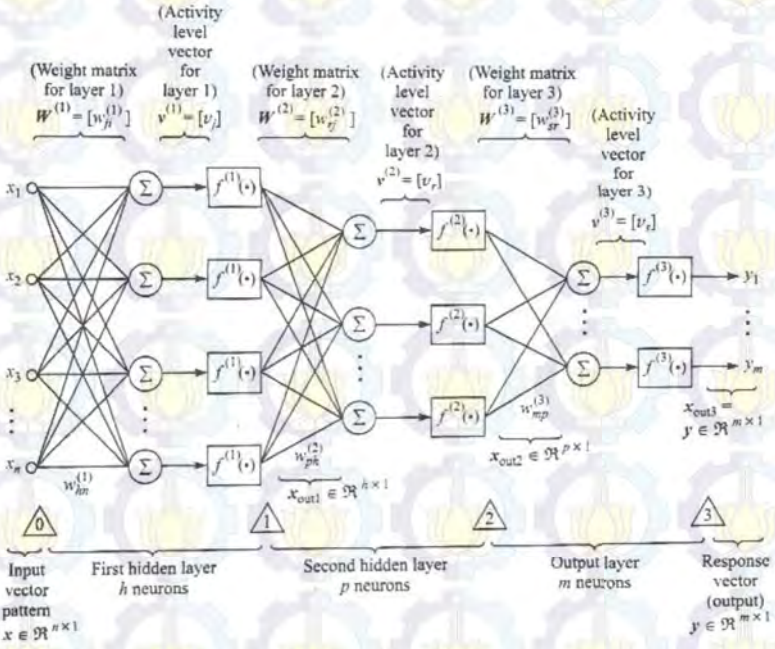
$$y_q = f_{lin}(v_q) = v_q \tag{2-8}$$

2.3.3 Backpropagation Neural Networks [13]

Backpropagation merupakan sebuah metode pembelajaran dalam *Multi Layer Perceptron* pada *Neural Networks*. Selama proses pembelajaran, *weight* dan bias diatur sedemikian rupa, untuk memperoleh

error yang minimum. Prinsip dasar dari algoritma *backpropagation* adalah *memperbaiki* bobot-bobot jaringan sehingga keluaran yang dihasilkan sesuai dengan batasan error minimal yang ditentukan.

Pada gambar 2.11, dimana i = 1, 2, . . . , n; j = 1, 2, . . . ,h; r = 1, 2, . . . , p; s = 1, 2, . . . , m; $f(\cdot)^{(1)}$ adalah fungsi aktivasi pada layer pertama, $f(\cdot)^{(2)}$ adalah fungsi aktivasi pada layer kedua, dan $f(\cdot)^{(3)}$ adalah fungsi aktivasi pada layer ketiga. Diasumsikan terdapat bias pada setiap neuron yang berada pada setiap layer dan juga terdapat matriks bobot yang terhubung antara *layer* berikutnya dengan *layer* setelahnya, yaitu $\boldsymbol{W}^{(\ell)}$ dengan $\ell = 1, 2, 3$. *Output* pada *layer* pertama, kedua, dan ketiga terdapat pada persamaan (2-9), (2-10) dan (2-11). Sedangkan respon akhir dari jringan (2-12) merupakan hasil dari subtitusi persamaan (2-9), (2-10) dan (2-11).



Gambar 2.11 Arsitektur tiga *layer perceptron*

$$\boldsymbol{x}_{out1} = \boldsymbol{f}^{(1)}[\boldsymbol{v}^{(1)}] = \boldsymbol{f}^{(1)}[\boldsymbol{W}^{(1)}\,\boldsymbol{x}] \quad (2\text{-}9)$$

$$\boldsymbol{x}_{out2} = \boldsymbol{f}^{(2)}[\boldsymbol{v}^{(2)}] = \boldsymbol{f}^{(2)}[\boldsymbol{W}^{(2)}\,\boldsymbol{x}_{out2}] \quad (2\text{-}10)$$

$$\boldsymbol{y} = \boldsymbol{x}_{out3} = \boldsymbol{f}^{(3)}[\boldsymbol{v}^{(3)}] = \boldsymbol{f}^{(3)}[\boldsymbol{W}^{(3)}\,\boldsymbol{x}_{out3}] \quad (2\text{-}11)$$

$$\boldsymbol{y} = \boldsymbol{f}^{(3)}[\boldsymbol{W}^{(3)}\boldsymbol{f}^{(2)}[\boldsymbol{W}^{(2)}\boldsymbol{f}^{(1)}[\boldsymbol{W}^{(1)}\,\boldsymbol{x}]]] \quad (2\text{-}12)$$

Pembelajaran dari model *neural networks* dengan metode *backpropagation* dimulai ketika *output* yang dihasilkan oleh *feedforward* tidak sesuai dengan yang *output* yang diharapkan. Perbandingan *output* ini didapatkan dengan cara menghitung error pada *layer output* menggunakan persamaan (2-13) dan pada se mua *layer* dengan persamaan (2-14). Setelah itu bobot nilai dan bias pada jaringan saraf tiruan akan diperbaharui dengan konstanta *learning rate* (μ) berdasarkan persamaan (2-15). 0

$$\delta_j^{(\ell)} = \left(d_{qh} - x_{out,j}^{(\ell)}\right) g\left(v_j^{(\ell)}\right) \quad (2\text{-}13)$$

$$\delta_j^{(\ell)} = \left(\sum_{h=1}^{\ell+1} \delta_h^{(s+1)}\, w_{hj}^{(\ell+1)}\right) g\left(v_j^{(\ell)}\right) \quad (2\text{-}14)$$

$$w_{ji}^{(\ell)}(k+1) = w_{ji}^{(\ell)}(k) + \mu\, \delta_j^{(\ell)}\, x_{out,j}^{(\ell-1)} \quad (2\text{-}15)$$

2.4 Euclidien distance[14]

Euclidiean distance merupakan sebuah dasar dari penentuan sebuah posisi atau lokasi relatif dari sebuah informasi jarak pada sebuah titik tertentu. Dimisalkan jarak antara titik p dan q pada bidang dimensi n mempunyai persamaan *euclidean distance* seperti berikut ini.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} \quad (2\text{-}16)$$

2.5 Kinect SDK [6]

Kinect Software Development Kit (SDK) merupakan sebuah *library* yang digunakan untuk melakukan pengembangan terhadap kamera Kinect. Sejak tahun 2011 SDK ini sudah mendukung sistem operasi Windows 7 dan kompatibel dengan *driver* perangkat Kinect.

Dengan menggunakan SDK, para pengembang dapat merancang berbagai macam jenis aplikasi dengan C++, C# atau Visual Basic menggunakan Visual Studio 2010. Kinect SDK juga mendukung kebutuhan untuk *image processing* seperti melakukan pengambilan, pengolahan serta penampilan data gambar. Interaksi antara manusia dan mesin melalui NUI juga dapat dilakukan. Pada umumnya SDK digunakan untuk pembuatan sebuah *control* pada suatu *game* yang berbasis Kinect.

2.6 Microsoft Visual Studio [15]

Microsoft Visual Studio merupakan sebuah *Integrated Development Environment* (IDE) yang dikembangkan oleh Microsoft. IDE ini digunakan untuk mengembankan atau membuat program-program komputer. *Platforms* yang digunakan pada Visual Studio yakni Windows API, Windows Forms, Windows Presentation Foundation, dan yang lainnya. Dengan adanya *Intellisense* yaitu sebuah editor kode yang dapat melengkapi kode, maka pengguna dapat dengan mudah dan cepat dalam menuliskan program yang akan dibuatnya. Selain pembuatan program yang mudah, Visual Studio menyediakan proses *debug* pada *souce-level* dan juga pada *machine-level*. Hal ini membuat proses *debug* menjadi lebih mudah. Bahasa pemrogaman yang didukung oleh Visual Studio diantaranya C, C++, C++/CLI, VB.Net, dan C#.

2.7 OpenCV [16]

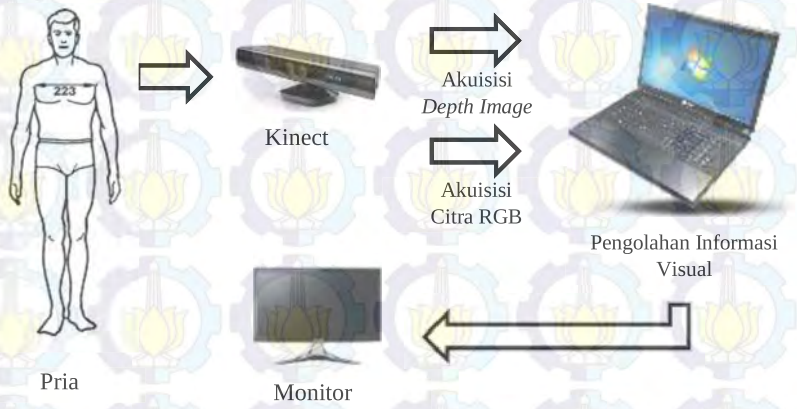
Open Source Computer Vision (Open CV) merupakan sebuah *library* program yang dikususkan untuk pemrogaman *computer vision* secara *real-time*. OpenCV dikembangkan oleh Intel dan saat ini ia didukung oleh Willow Garage. OpenCV dibuat dalam C++ sehingga penggunaan utamanya menggunakan bahasa C++, tetapi sekarang OpenCV mendukung *interface* selain C++, seperti Python, Java, C#, MATLAB. Hingga saat ini OpenCV dapat berjalan dengan baik pada sistem operasi Windows, Linux, Mac OS, Android, dan IOS.

2.8 MATLAB [17]

Matrix Laboratory (MATLAB) merupakan sebuah *environment* yang digunakan untuk melakukan penghitungan numerik berbasis matriks. MATLAB dapat melakukan manipulasi matriks, *plotting* suatu fungsi dan data, MATLAB dilengkapi dengan *interface* C, C++, Java, Fortran, dan Python. Untuk memudahkan penggunanya, MATLAB menyediakan berbagai macam jenis *toolbox* diantaranya *Curve Fitting Toolbox*, *Neural Network Toolbox*, *Control System Toolbox*, dan lain-lain.

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan pembuatan sistem perangkat lunak untuk melakukan pengolahan informasi visual. Sistem pengukuran lebar badan pria terdiri dari 4 komponen yang saling berhubungan diantaranya yaitu seorang pengguna, kamera Kinect, *processing unit*, pengolahan informasi visual, dan media komunikasi untuk pengguna. Hubungan antara komponen-komponen dapat di ilustrasikan pada gambar 3.1 berikut.



Gambar 3.1 Ilustrasi cara kerja sistem

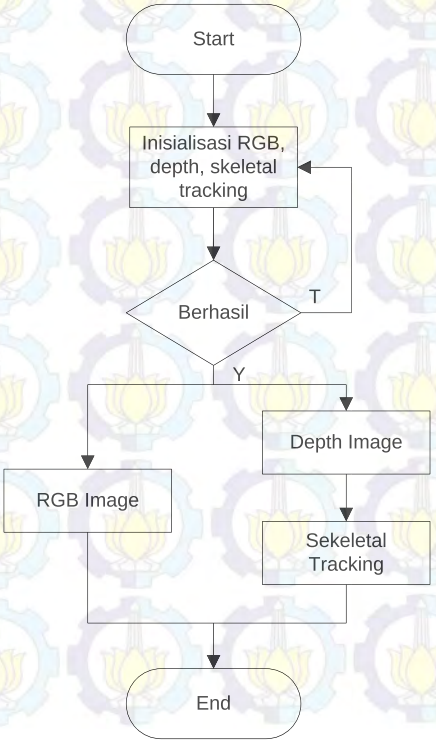
3.1 Prinsip Kerja

Menurut ilustrasi pada gambar 3.1, prinsip kerja dari sisten pengukuran lebar badan pria yang akan dirancang dimulai dari pengambilan informasi visual. Kinect merupakan perangkat yang digunakan untuk mengambil informasi visual dari pengguna. Informasi visual yang diambil diantaranya yaitu citra RGB dan *depth image.* Selanjutnya informasi visual tersebut akan dikirimkan menuju *processing unit*. Pada sistem yang akan dirancang, sebuah *notebook* digunakan sebagai *processing unit*. Informasi visual akan diolah untuk mendapatkan estimasi lebar badan dari pengguna. Dan terdapat program *inteface* yang akan membantu pengguna berinteraksi dengan sistem yang akan dirancang.

3.2 Kinect

Kamera Kinect merupakan perangkat yang digunakan untuk mengakuisisi citra RGB dan *depth image*, berikut ini merupakan spesifikasi dari kamera Kinect yang digunakan:

- Color Camera : 640 x 480 @ 30 fps
- Depth Camera : 320 x 240 @ 30 fps
- Jarak *Depth* Maksimal : 4 Meter
- Jarak *Depth Minimal* : 80 Sentimeter
- Jangkauan Pandangan Horizontal : 57 Derajat
- Jangkauan Pandangan Vertikal : 43 Derajat
- Sendi *Skeleton* yang disediakan : 20
- Jumlah *Skeleton Tracked* : 2



Gambar 3.2 Diagram alir inisialisasi kinect

Inisialisasi merupakan langkah awal dalam penggunaan kamera Kinect. Inisialisasi ini menentukan sensor-sensor yang akan digunakan kamera Kinect untuk melakukan pengambilan data. Disini penulis akan melakukanan inisialisasi untuk tiga jenis penggunaan kamera Kinect yaitu *color stream, depth stream,* dan *skeletal tracking*. *Color stream* merupakan *data stream* yang digunakan untuk melakukan pengambilan data citra RGB dengan cara, `NUI_INITIALIZE_FLAG_USES_COLOR` *Depth stream* juga termasuk *data stream* yang digunakan untuk mengambil data, tetapi data yang diambil disini yaitu berupa data kedalaman. Proses inisialisai yaitu `NUI_INITIALIZE_FLAG_USES_DEPTH`. Sedangkan *skeletal tracking* digunakan untuk melakukan identfikasi pada *frame* apakah terdapat data yang menyerupai *skeletal*. *Flag* yang digunakan yaitu `NUI_INITIALIZE_FLAG_USES_SKELETON`. Untuk menggunakan tiga macam pengambilan data secara sekaligus, maka *flag* inisialisasi ini dapat digabungkan dengan *bitwise* OR sehingga menjadi:

```
hr= m_pNuiSensor -> NuiInitialize (NUI_INITIALIZE_FLAG_USES_COLOR|
NUI_INITIALIZE_FLAG_USES_DEPTH | NUI_INITIALIZE_FLAG_USES_SKELETON
);
```

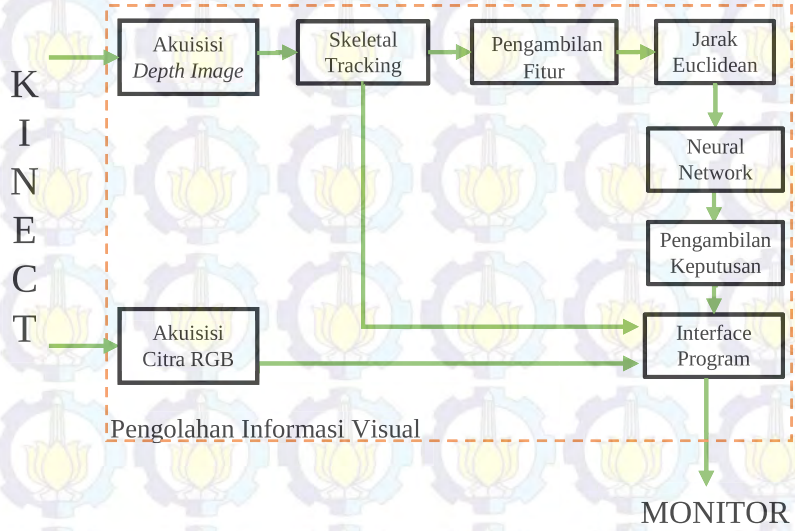
3.3 Pengolahan Informasi Visual

Untuk menentukan estimasi lebar badan dari penggunanya, maka sistem perlu mengambil informasi-informasi visual yang berkaitan dengan penggunanya. Langkah-langkah sistem untuk melakukan estimasi lebar badan pengguna menurut gambar 3.2 adalah seperti berikut:

a. Kinect menangkap informasi citra RGB, data *depth* setiap *frame*. Bila ada orang pada *depth frame,* maka Kinect akan melakukan *skeletal tracking*.
b. Citra RGB dan *skeletal traking* akan diperlihatkan ke *user* melalui program *interface.*
c. *Skeletal tracking* akan memberikan posisi setiap sendi dalam koordinat *skeleton space*.
d. Fitur yang diambil adalah jarak *depth* dari beberapa sendi yang dipilih.
e. Berdasarkan posisi sendi bahu kanan dan bahu kiri, jarak antar bahu dapat diketahui dengan Jarak Euclidean.
f. Data *depth* yang didapat menjadi masukkan untuk jaringan saraf tiruan yang akan menentukan regresi untuk setiap masukkan yang

diberikan yang pada akhirnya dapat memberikan estimasi lebar badan dari pengguna.

g. Pengambilan keputusan akan memberikan saran kategori ukuran baju berdasar hasil estimasi lebar badan.
h. Perkiraan lebar badan yang didapatkan dari proses jaringan saraf tiruan akan disampaikan ke user melalui program *interface*.



Gambar 3.3 Diagram blok pengolahan informasi visual

3.3.1 Akuisisi Depth *Image*

Langkah pertama untuk visualisasi informasi jarak dari *depth image* yaitu dengan cara melakukan penguncian *depth frame* agar data *depth stream* tidak mengalami perubahan saat dilakukan pembacaan. Data *depth stream* akan dimasukkan pada `depthh` melalui `cvSetData(depthh, pBuff, kuncirect.Pitch)`. `depthh` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame depth stream* sedangkan `kuncirect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*. Setelah dilakukan pembacaan data maka akan divisualkan dengan `cvShowImage("Depth Image",depthh)` serta *frame* dibuka untuk pengambilan data berikutnya.

Gambar 3.4 *Depth image*

```
if (kuncirect.Pitch!=0)
{
  int minDepth = (modenear ? NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
    NUI_IMAGE_DEPTH_MINIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
  int maxDepth = (modenear ? NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
    NUI_IMAGE_DEPTH_MAXIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
  BYTE* runwarna = m_depthRGBX;
    const NUI_DEPTH_IMAGE_PIXEL* BuffRC;
    const NUI_DEPTH_IMAGE_PIXEL* BuffLC;
    const NUI_DEPTH_IMAGE_PIXEL* BuffRS;
    const NUI_DEPTH_IMAGE_PIXEL* BuffLS;
    const NUI_DEPTH_IMAGE_PIXEL* BuffBelly;
    const NUI_DEPTH_IMAGE_PIXEL* BuffChest;
    const NUI_DEPTH_IMAGE_PIXEL* RunBuffer = reinterpret_cast <
      const NUI_DEPTH_IMAGE_PIXEL*>(kuncirect.pBits);
    const NUI_DEPTH_IMAGE_PIXEL* EndBuffer = RunBuffer +
      (cDepthWidth*cDepthHeight) -1;
    int selisih = 0;
    selisih = EndBuffer-RunBuffer;
    while (RunBuffer<EndBuffer)
    {
      USHORT depth= RunBuffer->depth;
      BYTE intensitasnya = static_cast<byte>(depth >= minDepth &&
        depth <= maxDepth ? depth % 255:0);
      *(runwarna++) = intensitasnya;
      *(runwarna++) = intensitasnya;
```

```
      *(runwarna++) = intensitasnya;
      ++runwarna;
      ++RunBuffer;
    }
    RunBuffer = RunBuffer-selisih;
    USHORT * pBuff = (USHORT*)m_depthRGBX;
    cvSetData(depthh,pBuff,kuncirect.Pitch);
  }
cvShowImage("Depth Image",depthh);
tekstur->UnlockRect(0);
tekstur->Release();
ReleaseFrame:
m_pNuiSensor-> NuiImageStreamReleaseFrame (m_pDepthStreamHandle,
  &imageFrame);
cvWaitKey(1);
```

3.3.2 Akuisisi Citra RGB

Gambar 3.5 Citra RGB

Untuk menampilkan citra RGB maka *frame color stream* akan dikunci agar saat pengambilan data, data pada *color frame* tidak berubah. Lalu data *color stream* dimasukkan kedalam variabel `color`, dengan cara `cvSetData(color, pBuffer, LockedRect.Pitch)`, dimana `color` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame color stream* sedangkan `LockedRect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*.

```
if (LockedRect.Pitch != 0)
{
```

```
  BYTE * pBuffer = (BYTE*) LockedRect.pBits;
  cvSetData(color, pBuffer, LockedRect.Pitch);
}
cvShowImage("Color Image", color);
tekstur->UnlockRect(0);
m_pNuiSensor->NuiImageStreamReleaseFrame(m_pColorStreamHandle,
  &imageFrame);
cvWaitKey(1);
```

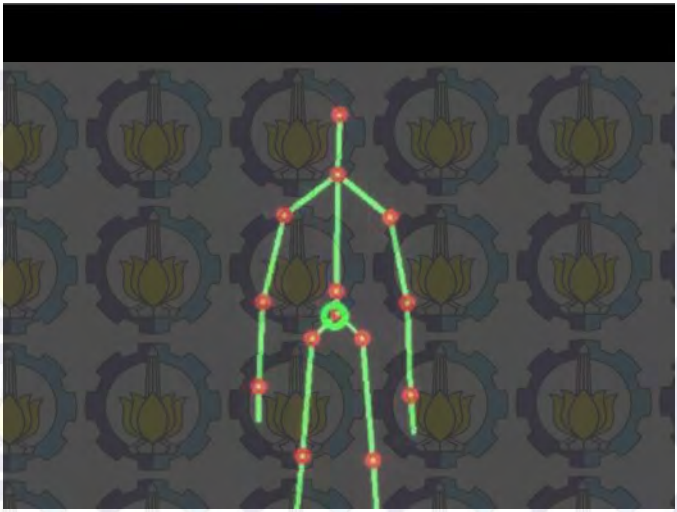
3.3.3 Skeletal Tracking

Pendeteksian *skeletal* memanfaatkan *library* yang berasal dari Kinect. *Library* ini akan mendeteksi kondisi ada atau tidaknya *skeletal* secara. *Skeletal tracking* dimulai dengan memanggil fungsi `void drawSkeleton(IplImage* image)`. Variabel `image` merupakan sebuah data citra. Selanjutnya sensor Kinect akan mendeteksi ada atau tidaknya manusia pada *frame*. Ketika pada *frame* terdeteksi ada manusia maka *skeletal* akan digambarkan secara penuh (20 sendi) pada *frame* dan dapat mengikuti (*tracking*) gerakan manusia.

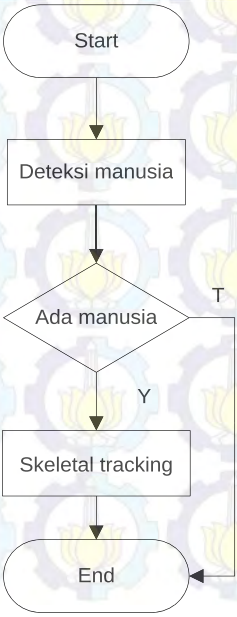
Untuk proses visualisasi dan memudahkan dalam proses berikutnya, koordinat sendi pada *skeletal tracking* akan di konversi dari sistem koordinat *skeleton space* menjadi sistem koordinat *color space*. konversi koordinat dilakukan dengan cara:

```
NuiTransformSkeletonToDepthImage (sendi[jointFrom],
&SenDepthX[jointFrom], &SenDepthY[jointFrom],
NUI_IMAGE_RESOLUTION_640x480);
m_pNuiSensor->
NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(
  NUI_IMAGE_RESOLUTION_640x480, NUI_IMAGE_RESOLUTION_640x480,0,
  (LONG)SenDepthX[jointFrom], (LONG)SenDepthY[jointFrom], 0,
  &SenPosX[jointFrom],&SenPosY[jointFrom]);
```

`NuiTransformSkeletonToDepthImage` akan mengubah posisi sendi *skeleton space* menjadi *depth space* pada resolusi 640x480 dengan nama variabel `SenDepthX[]` untuk sumbu x dan `SenDepthY[]` untuk sumbu y. Setelah itu *depth space* dikonversi menjadi *color space* melalui `NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution` dari resolusi 640x480 dengan resolusi tetap 640x480 dimana variabel `SenPosX[]` untuk sumbu x dan `SenPosY[]`untuk sumbu y.



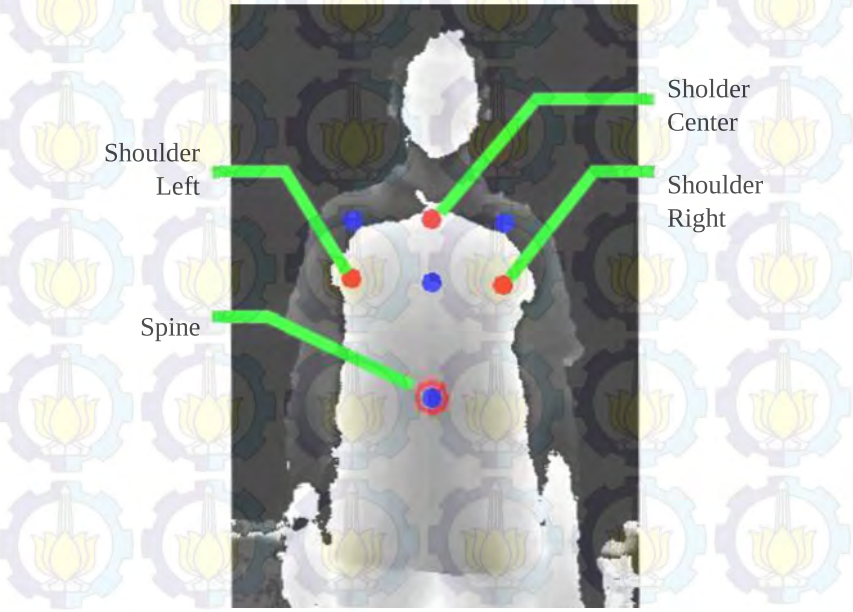
Gambar 3.6 Hasil *Skeletal tracking*



Gambar 3.7 Diagram alir *skeletal tracking*

3.3.4 Pengambilan Fitur

Untuk pengambilan fitur, maka diperlukan beberapa informasi jarak pada *depth image*. informasi tersebut didapat dari empat sendi *skeletal tracking* diantaranya yaitu *spine, shoulder center, shoulder left,* dan *shoulder right*. Bila dilihat pada gambar 3.7 terdapat tiga lingkaran merah, tiga lingkaran biru, dan satu lingkaran biru dengan lingkaran merah disekelilingnya. Tiga lingkaran merah menandakan posisi sendi secara *default* dari kamera Kinect. Tiga sendi ini tidak akan diambil informasi *depth*-nya. Selanjutnya tiga lingkaran biru ini merupakan posisi sendi-sendi bahu dan dada yang mengalami penyesuaian. Pada sendi *shoulder left* dan *right* memakai sumbu Y dari sendi *shoulder center*. Sedangkan pada sendi *shoulder center* memakai sumbu Y dari *shoulder left* dan *right*. Hal ini dilakukan untuk meningkatkan kestabilan dari informasi *depth* yang diambil. Sedangkan satu lingaran biru dengan lingkaran merah disekelilingnya tidak mengalami penyesuaian dan akan diambil informasi *depth*-nya.



Gambar 3.8 Posisi sendi-sendi (*default*) pada *depth image*

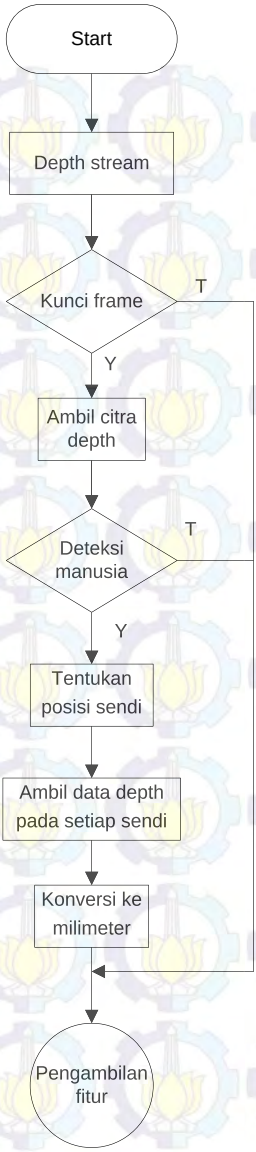
Setelah menentukan posisi sendi yang akan diambil informasi *depth*-nya, konversi data *depth* menjadi milimeter dilakukan untuk mengetahui jarak dari Kinect terhadap permukaan dimana posisi sendi tersebut berada. Untuk mendapatkan jarak antara kamera dengan permukaan posisi sendi tersebut dilakukan saat penguncian *frame depth stream*. Berikut ini cara untuk mendapatkan jarak dalam satuan milimeter untuk setiap posisi sendi:

```
BuffChest = BuffChest + Chestpx;
BuffBelly = BuffBelly + Bellypx;
BuffLS = BuffLS + LSpx;
BuffRS = BuffRS + RSpx;
bellydepth = BuffBelly->depth;
chestdepth = BuffChest->depth;
LSdepth = BuffLS->depth;
RSdepth = BuffRS->depth;
```

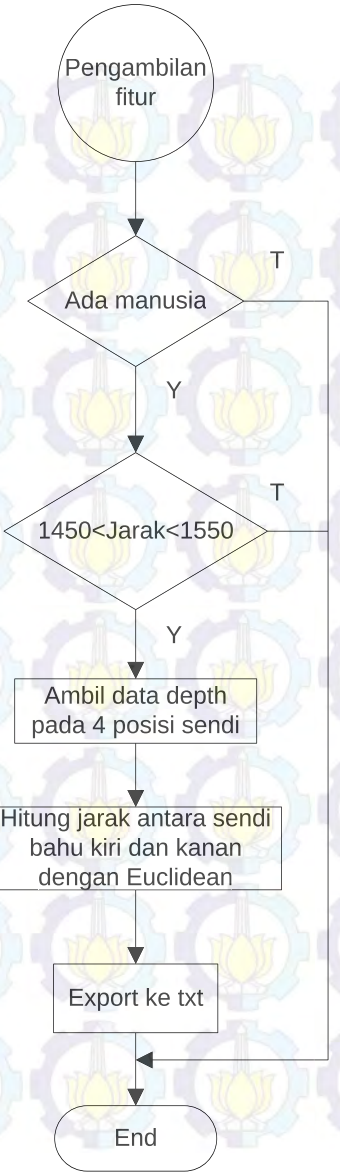
`BuffChest`, `BuffBelly`, `BuffLS`, dan `BuffRS` merupakan variabel piksel yang menyimpan informasi jarak. Sedangkan `Chestpx` (*shoulder center*), `Bellypx` (*spine*), `LSpx` (*shoulder left*), dan `RSpx` (*shoulder right*) merupakan posisi pixel dimana sendi itu berada. Dan hasil pengukuran jarak untuk setiap sendi teradapat pada `bellydepth`, `chestdepth`, `LSdepth`. Dari hasil konversi tersebut, maka fitur-fitur yang digunakan untuk melakukan perkiraan lebar badan akan diproses oleh sistem. Fitur-fitur tersebut diantaranya:

a. Jarak antara Kinect dengan permukaan bahu kiri.
b. Jarak antara Kinect dengan permukaan bahu kanan.
c. Selisih jarak antara permukaan bahu dengan dada.
d. Selisih jarak antara permukaan bahu dengan perut.

Menurut diagram alir pengambilan fitur pada gambar 3.8, langkah pertama untuk pengambilan fitur ini yaitu mendeteksi ada atau tidaknya manusia pada *frame*. Selanjutnya menentukan jarak pengguna dengan kamera akan agar berada pada 1450 mm hingga 1550 mm. Dan yang terakhir menjaga posisi badan lurus dengan kamera dengan cara membandingkan *depth* antara sendi bahu kanan dan bahu kiri sehingga pengambilan fitur dimulai dan di *export* dengan format txt.



Gambar 3.9 Diagram alir pengambilan fitur



Gambar 3.10 Diagram alir pengambilan fitur

3.3.5 Jarak Euclidean

Untuk memperkirakan estimasi lebar badan dari pengguna maka jarak antara sendi pada bahu kanan dan sendi pada bahu kiri akan diestimasi terlebih dahulu. Sendi-sendi yang didapat dari *skeletal tracking* berada dalam koordinat tiga dimensi. Persamaan Euclidean dari titik p ke titik q pada ruang tiga dimensi yaitu:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 - (p_2 - q_2)^2 - (p_3 - q_3)^2} \qquad (3\text{-}1)$$

```
FLOAT diff_x = LS.x - RS.x;
FLOAT diff_y = LS.y - RS.y;
FLOAT diff_z = LS.z - RS.z;
lebar_s = (sqrt(pow(diff_x,2)+pow(diff_y,2)+pow(diff_z,2))*100);
```

3.3.6 Neural Network

Artificial Neural Networks dalam dunia *machine learning* sering disebut dengan *Neural Networks* atau Jaringan Saraf Tiruan (JST) merupakan sebuah metode pembelajaran statistik berdasarkan sistem saraf pada bagian otak. JST dapat melakukan *mapping* antara masukkan dan keluaran ataupun mencari korelasi antar keduanya. Korelasi didapatkan dari pasangan masukkan dan keluaran yang telah dilatihkan.

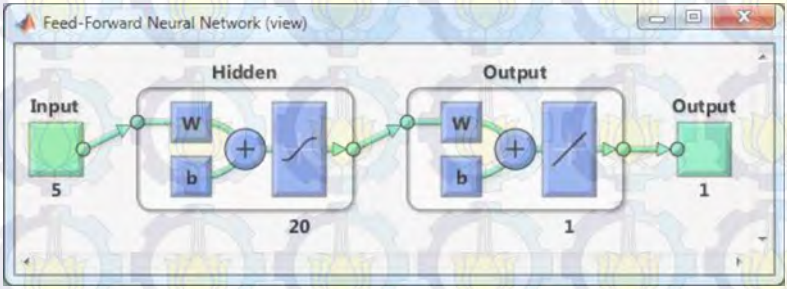
Penggunaan JST karena kemampuannya untuk melakukan generalisasi pada keluaran, meskipun informasi-informasi yang diterima tidak ada pada informasi-informasi yang telah dilatihkan. Selain itu JST juga dapat menentukan tren data yang sangat kompleks yang tidak terlihat oleh pengamatan manusia.

JST sangat membantu bila pada suatu sistem membutuhkan *multiple input* dan juga *multiple output*. Bila sistem menggunakan *single input* dan *single output* maka metode regresi sudah cukup untuk di implementasikan pada sistem. Pada Tugas Akhir ini JST digunakan untuk melakukan pembelajaran agar dapat memperkirakan lebar badan pengguna. Pembelajaran jaringan saraf tiruan menggunakan MATLAB. Penulis menggunakan MATLAB untuk mempercepat proses pembelajaran dan menghindari *overfit* yang dapat terjadi[18]. *Mean Squared Error* (MSE) digunakan untuk mengetahui performa dari jaringan saraf tiruan.

Setelah pengambilan *fitur* di *eksport* dalam bentuk txt, kemudian *file* tersebut akan menjadi sumber data pembelajaran untuk MATLAB.

Jaringan saraf tiruan yang diterapkan disini yaitu *multi layer percetpron backpropagation* menggunakan 5 *node input layer*, 20 *node hidden layer* 1, dan 1 *node output layer*. Fungsi aktivasi yang digunakan yaitu bipolar sigmoid pada *hidden layer* dan fungsi aktivasi linier pada *output layer*. Fungsi aktivasi yang digunakan bertujuan untuk mendapatkan tren data dari estimasi lebar badan. Adapun masukkan yang diberikan yaitu:

- Jarak kedalaman (*depth*) antara permukaan bahu kiri pengguna dengan Kinect.
- Jarak kedalaman (*depth*) antara permukaan bahu kanan pengguna dengan Kinect.
- Selisih kedalaman (*depth*) antara permukaan bahu pengguna dengan permukaan dada pengguna.
- Selisih kedalaman (*depth*) antara permukaan bahu pengguna dengan permukaan perut pengguna.
- Estimasi lebar bahu pengguna berdasarkan metode jarak Euclidean.



Gambar 3.11 Konfigurasi MLP Backpropagation

Setelah MATLAB selesai melakukan pembelajaran maka nilai bobot dan bias antar *node* di *eksport* ke dalam *file* txt dan siap digunakan untuk tahapan estimasi pengukuran lebar badan. Estimasi pengukuran lebar badan atau bagian *feedforward* dari *multi layer backpropagation* di terapkan pada C++ dimulai dengan *import* data-data txt diantaranya:

a. Input.txt, sebagai *input database* yang nantinya akan digunakan untuk proses normalisasi.
b. Output.txt, sebagai *output database* yang nantinya akan digunakan untuk proses normalisasi.

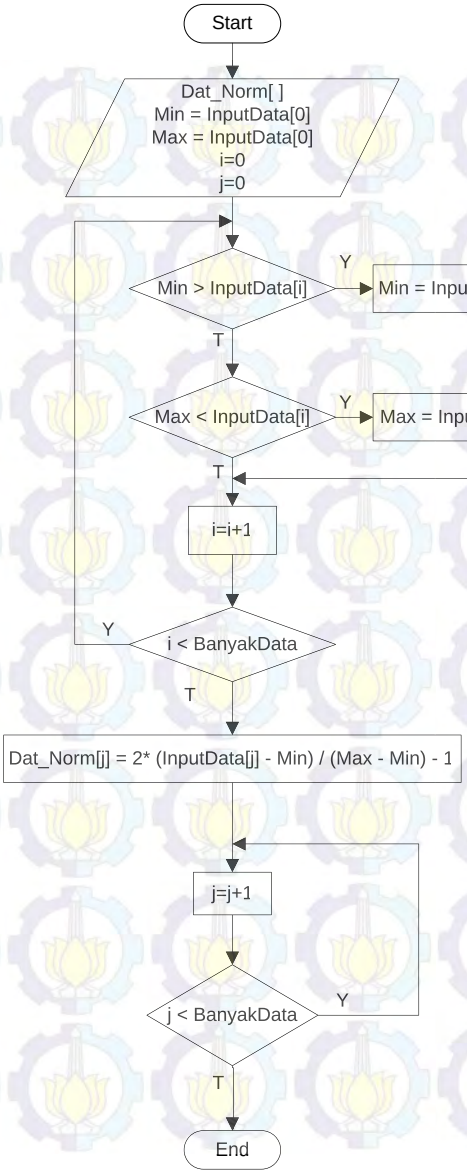
c. w1.txt, bobot nilai setiap node yang berada pada *input layer* dan *hidden layer*.
d. w2.txt, bobot nilai setiap node yang berada pada *hidden layer* dan *output layer*.
e. bias1.txt, nilai bias yang berada pada setiap node pada *hidden layer*.
f. bias2.txt, nilai bias yang berada pada setiap node pada *output layer*.

Untuk meningkatkan performa pebelajaran dari jaringan saraf tiruan, perlu adanya proses normalisasi pada bagian *input*, dan denormalisasi pada bagian *output*. Metode normalisasi (3-2) dan denormalisasi (3-3) yang digunakan yaitu metode min-max. Pada proses normalisasi, nilai minimal dan juga nilai maksimal yang dicari merupakan nilai yang berada pada satu node *input* yang sama, bukan nilai minimal dan maksimal dari lima node *input*.

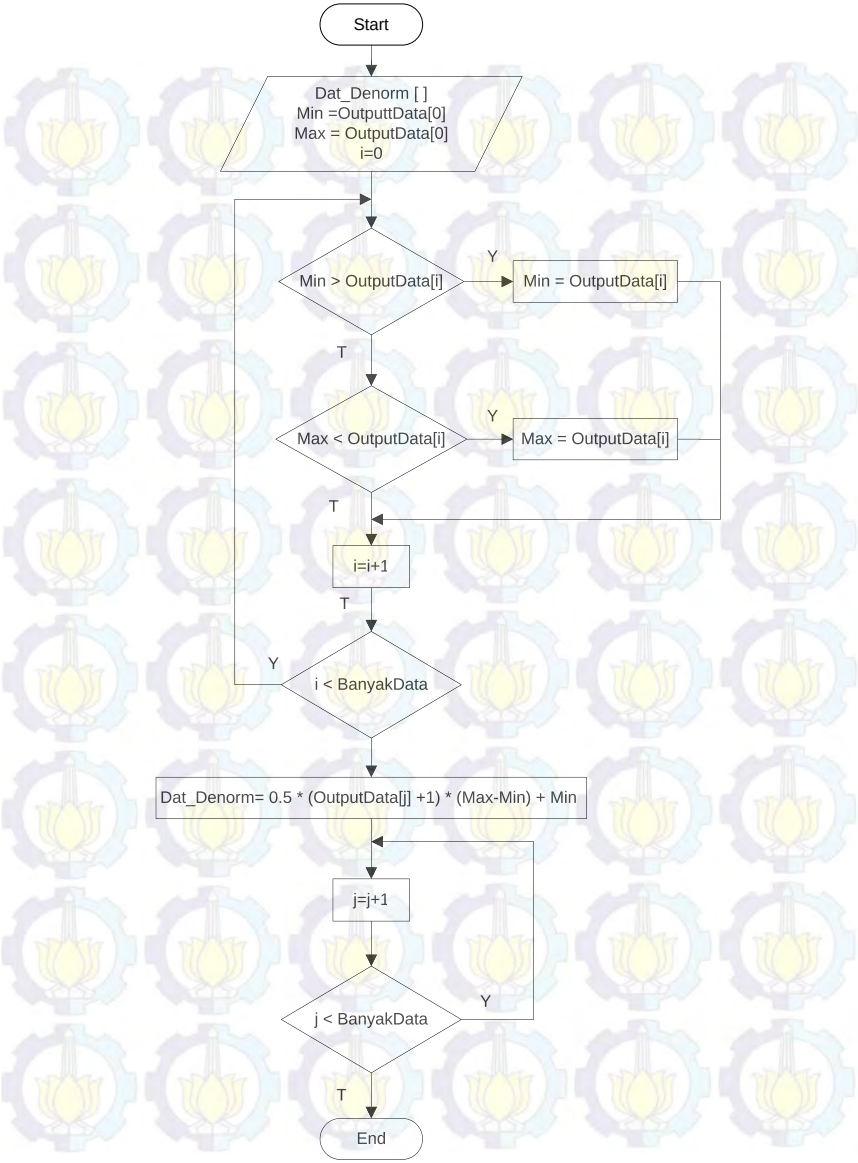
$$y = \frac{(y_{max} - y_{min})(x_{max} - x_{min})}{(x_{max} - x_{min})} - y_{min} \quad (3\text{-}2)$$

$$x = \frac{(x_{max} - x_{min})(y_{max} - y)}{(y_{max} - y_{min})} - x_{min} \quad (3\text{-}3)$$

Dimana:
y_{max} = Nilai maksimum hasil normaliasi
y_{min} = Nilai minimum hasil normaliasi
x_{max} = Nilai maksimum data
x_{min} = Nilai minimum data



Gambar 3.12 Diagram alir normalisasi



Gambar 3.13 Diagram alir denormalisasi

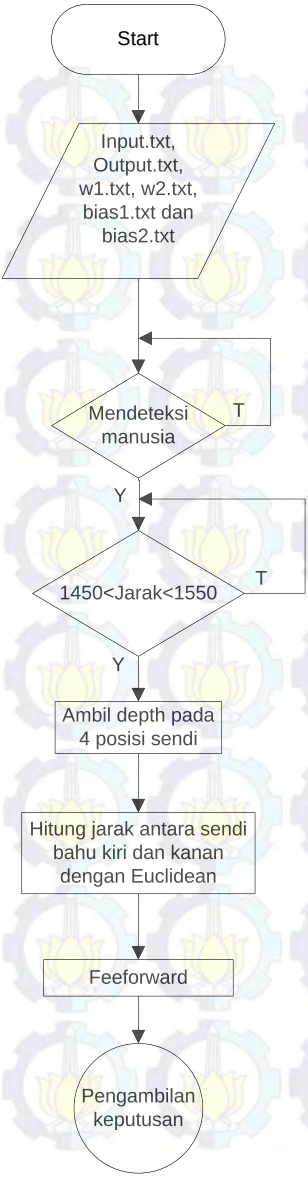
3.3.7 Pengambilan Keputusan

Estimasi lebar badan dilakukan dengan mengambil fitur-fitur dari pengguna yang berdiri di depan kamera dilanjutkan dengan proses *feedforward* MLP. Proses pertama pada *feedforward* yaitu *import* 6 *files* yang telah disediakan, dilanjutkan dengan pengambilan fitur, lakukan normalisasi terhadap fitur, proses dengan *feedforward*, denormalisasi hasil *feedforward* dan yang terakhir lakukan pengambilan keputusan untuk menentukan kategori ukuran baju berdasarkan hasil estimasi lebar badan. Dimana baju dikategorikan menjadi 4 yaitu S, M, L XL dan - (tidak ada ukuran baju yang sesuai), sesuai dengan tabel 3.1 berikut.

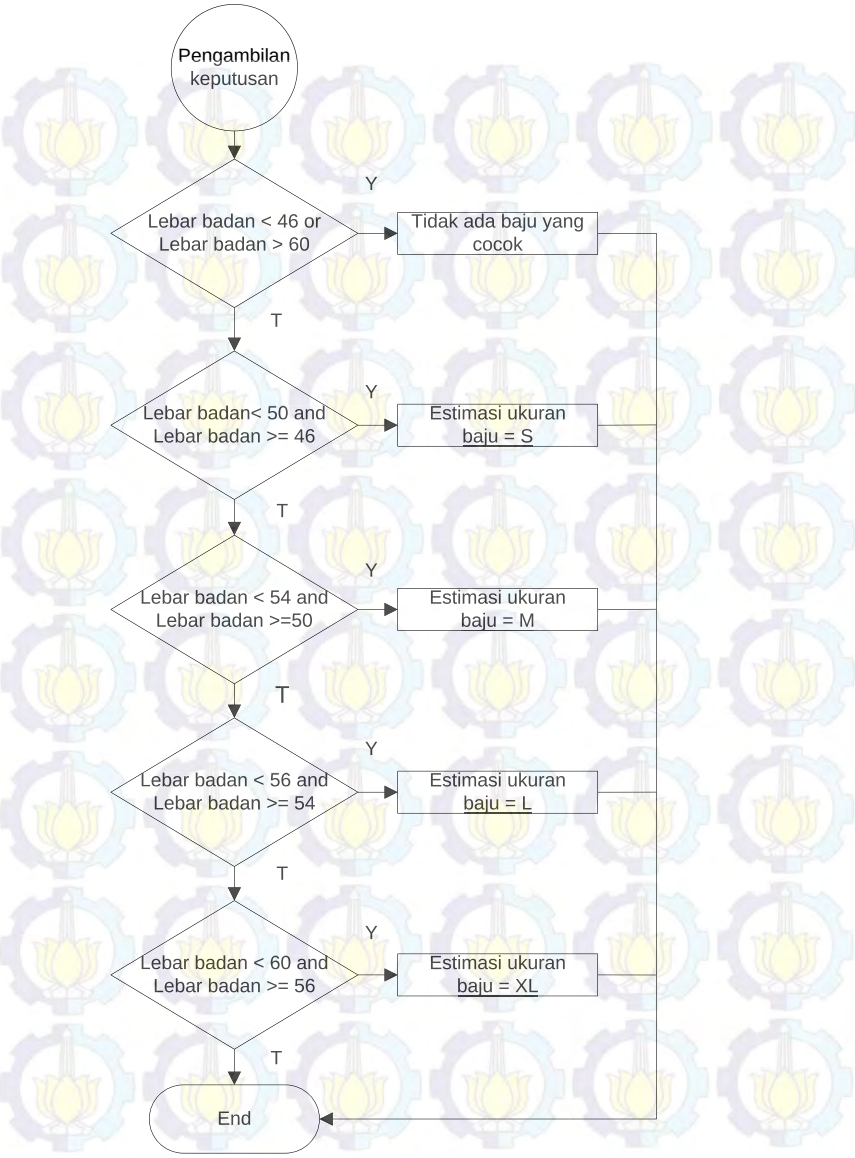
Tabel 3.1 Tabel kategori ukuran baju

NO	Kategori Baju	Lebar Badan Minimal (cm)	Lebar Badan Maksimal (cm)
1	S	46	50
2	M	50	54
3	L	54	56
4	XL	56	60
5	-	Kurang dari 46	Lebih dari 60

Gambar 3.14 Contoh kategori ukuran baju



Gambar 3.15 Diagram alir proses estimasi ukuran baju



Gambar 3.16 Diagram alir proses estimasi badan

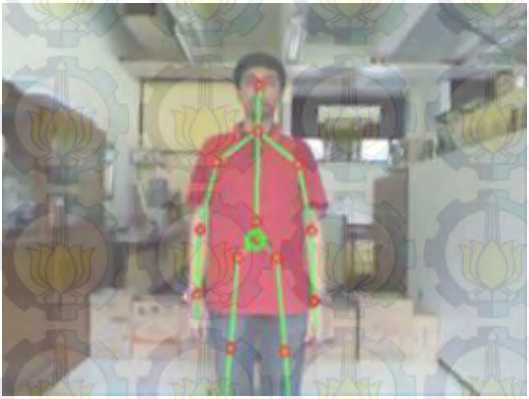
3.3.8 *Interface Program*

Program *interface* merupakan sebuah antarmuka antara sistem yang dirancang dengan pengguna. Melalui *interface* ini, pengguna dapat berinteraksi dengan sistem dan melihat tampilan dirinya disertai dengan *skeletal tracking*. Untuk kepentingan pengukuran maka disertakan indikator jarak, indikator lurus atau tidaknya badan terhadap kamera, estimasi jarak antar bahu, estimasi lebar badan, ukuran baju yang disarankan dan status dari sistem. Indikator jarak akan merubah gambar yang ditangkap oleh kamera Kinect menjadi merah bila pengguna berdiri diluar dari jarak yang dianjurkan (1500 mm. Selain itu sistem ini juga dilengkapi dengan NUI untuk melakukan pengukuran ulang dengan cara menempelkan tangan kanan ke dada.

Gambar 3.17 Tampilan program *interface*

Gambar yang ditampilkan pada *interface* merupakan gabungan antara citra RGB dan *skeletal tracking*. Untuk menampilkan citra RGB dan *skeletal tracking* maka *frame color stream* akan dikunci agar saat pengambilan data, data pada *color frame* tidak berubah. Lalu data *color stream* dimasukkan kedalam variabel `color`, dengan cara `cvSetData(color, pBuffer, LockedRect.Pitch)`, dimana `color` merupakan sebuah *header* dari *array*. `pBuffer` adalah data dari *frame color stream* sedangkan `LockedRect.Pitch` merupakan panjang keseluruhan baris dalam *bytes*. Selanjutnya `drawSkeleton(color)` dipanggil untuk proses *skeletal tracking*.

```
if (LockedRect.Pitch != 0)
{
    BYTE * pBuffer = (BYTE*) LockedRect.pBits;
    cvSetData(color, pBuffer, LockedRect.Pitch);
    drawSkeleton(color);
    pictureBox1->Image  = gcnew System::Drawing::Bitmap(
    color->width,color->height,color->widthStep,System::
    Drawing::Imaging::PixelFormat::Format32bppRgb,(System::IntPtr)
    color->imageData);
    pictureBox1->Refresh();
}
```



Gambar 3.18 Hasil perpaduan citra RGB dan *skeletal tracking*

3.4 Layar Monitor

Untuk tampilan *display* ke pengguna digunakan sebuah layar monitor ViewSonic Va1601w 15,5 inch dengan resolusi 1280x720.

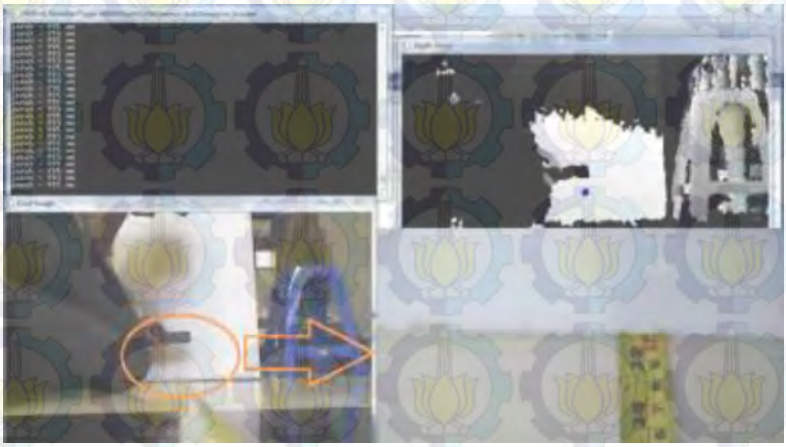
Gambar 3.19 Layar Monitor ViewSonic

BAB IV
PENGUJIAN DAN ANALISIS

Pengujian sistem dimaksudkan untuk mengetahui hasil dari sistem yang telah dirancang. Pengujian ini dimulai dengan pengukuran jarak dengan kamera Kinect. Selanjutnya pengujian akan terbagi menjadi dua yakni pada jarak optimal sistem dan pada jarak yang tidak optimal. Pengujian tersebut diantaranya yaitu pengukuran jarak antar bahu dengan metode jarak Euclidean, estimasi lebar badan dengan *Neural Networks*, dan penentuan kategori baju. Pengujian tersebut dilakukan untuk mengetahui pengaruh jarak pengguna terhadap hasil pengukuran sistem.

4.1 Pengukuran Jarak dengan Sensor Kinect

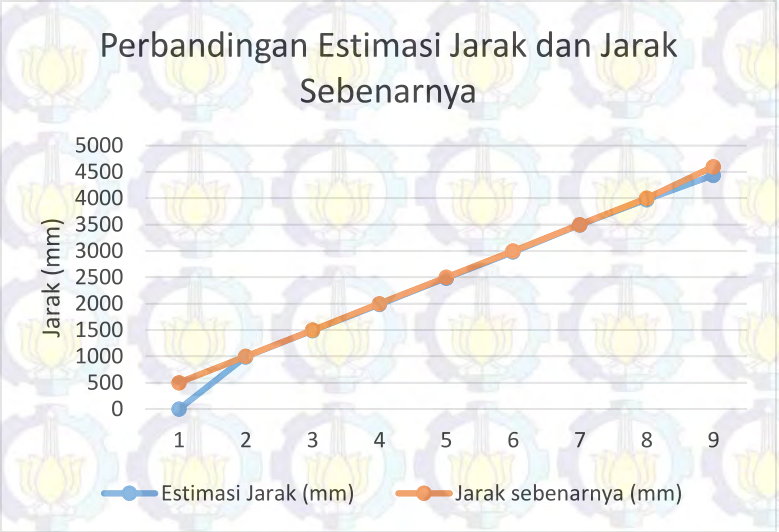
Dengan menggunakan informasi dari *Kinect IR Sensors* maka jarak antara suatu obyek dengan Kinect dapat diestimasi. Jarak tersebut tersebut umumnya disebut dengan *depth.* Data *depth* yang didapatkan dari sensor Kinect mempunyai satuan milimeter. Pengujian jarak ini dilakukan dengan cara membandingkan data *depth* yang terukur oleh sensor Kinect dengan jarak yang sebenarnya. Pengukuran jarak dilakukan pada jarak 500 mm hingga 4.500 mm dengan interval 500 mm. Pemilihan rentang jarak antara 500 mm hingga 4.500mm merupakan jarak optimal yang dapat destimasi oleh Kinect.



Gambar 4.1 Pengukuran jarak menggunakan data *depth* sensor Kinect

Tabel 4.1 Perbandingan estimasi jarak dan jarak sebenarnya

No	Estimasi Jarak (mm)	Jarak sebenarnya (mm)	Error Absolut (%)
1	0	500	100
2	993	1000	0,7
3	1489	1500	0,73
4	1986	2000	0,7
5	2487	2500	0,52
6	2980	3000	0,66
7	3493	3500	0,2
8	3975	4000	0,62
9	4434	4600	3,60



Gambar 4.2 Grafik perbandingan estimasi jarak dan jarak sebenarnya

Berdasarkan tabel 4.1 dapat dilihat bahwa sensor Kinect tidak dapat mendeteksi jarak 500 mm. Jarak minimal yang dapat diestimasi oleh sensor Kinect yaitu 1000 mm. Sensor Kinect tidak dapat mendeteksi jarak 500 mm sehingga memberikan estimasi jarak 0 mm. Sedangkan antara

jarak 1000 mm hingga 4000 mm perbandingan jarak antara hasil estimasi oleh sensor Kinect dengan jarak sebenarnya tidak mengalami perbedaan yang signifikan. Terlihat pada tabel 4.1 bahwa error absolut pada jarak tersebut kurang dari 1%. Namun pada jarak 4500 mm hasil estimasi jarak oleh sensor Kinect mengalami perbedaan yang cukup jauh dibandingkan jarak-jarak yang sebelumnya dan terdapat error abslut 3,6%. Hal ini dikarenakan pengukuran jarak dilakukan diluar batas pengukuran optimal yang dapat dilakukan oleh sensor Kinect.

4.2 Pelatihan Data pada Jarak 1500 Milimeter

4.2.1 Pengukuran Jarak Antar Bahu

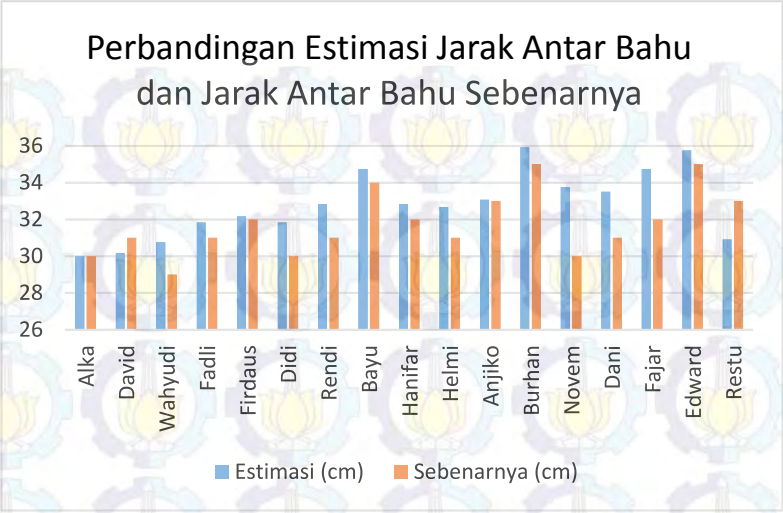
Pengujian estimasi jarak antar bahu dengan metode jarak Euclidean, yakni menentukan jarak antara dua buah titik[14]. Untuk estimasi jarak antar bahu, titik yang dihitung yakni jarak Euclidean antara sendi bahu kanan dan sendi bahu kiri yang terdeteksi oleh *skeletal tracking*. Nantinya hasil pengujian ini akan dibandingkan dengan pengukuran jarak antar bahu yang sebenarnya. Untuk pengukuran jarak antar bahu yang sebenarnya dilakukan dengan cara mengukur dengan pita meter seperti yang terdapat pada gambar 2.1. Pengujian ini dilakukan pada 17 orang dan jarak antara kamera Kinect dengan subjek pengujian yaitu sebesar 1500 mm dengan batas toleransi ± 15 mm. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 1.

Gambar 4.3 Hasil dari estimasi jarak antara bahu

Tabel 4.2 Perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Alka	30,00	30	0,00
2	David	30,17	31	2,69
3	Wahyudi	30,75	29	6,03
4	Fadli	31,83	31	2,69
5	Firdaus	32,17	32	0,52
6	Didi	31,83	30	6,11
7	Rendi	32,83	31	5,91
8	Bayu	34,75	34	2,21
9	Hanifar	32,83	32	2,60
10	Helmi	32,67	31	5,38
11	Anjiko	33,08	33	0,25
12	Burhan	35,92	35	2,62
13	Novem	33,75	30	12,50
14	Dani	33,50	31	8,06
15	Fajar	34,75	32	8,59
16	Edward	35,75	35	2,14
17	Restu	30,92	33	6,31

Pada tabel 4.2 terlihat bahwa perbandingan estimasi jarak antar bahu dengan jarak antar bahu sebenarnya terdapat error absolut yang cukup besar pada beberapa orang yaitu hingga 12,5%. Sedangkan rata-rata dari error absolut yang didapatkan yaitu 4,38% Perbedaan hasil estimasi jarak antar bahu dengan jarak antar bahu yang sebenarnya disebabkan karena penggunaan pakaian yang beragam serta pemakaian pakaian yang cenderung lebih besar dari ukuran badan yang semestinya.



Gambar 4.4 Grafik perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

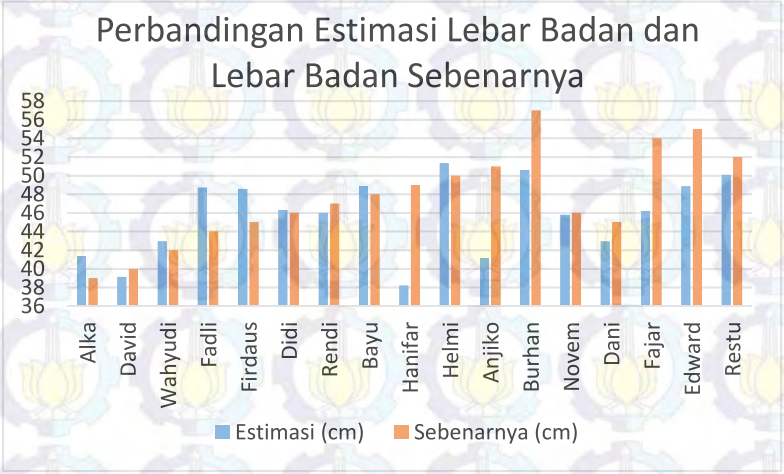
4.2.2 Estimasi Lebar Badan

Estimasi lebar badan dapat dilakukan setelah melatihkan 60 set data uji dari 12 subjek pengujian. Setiap subjek mempunyai lima set data uji. Pengambilan lima set data uji ini dimaksudkan untuk mengurangi kesalahan estimasi lebar badan akibat pergerakkan kecil seperti bernapas.

Tabel 4.3 Perbandingan estimasi lebar badan dan lebar badan sebenarnya

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Alka	41,36	39	3,10
2	David	39,14	40	1,67
3	Wahyudi	42,95	42	0,09
4	Fadli	48,72	44	3,20
5	Firdaus	48,59	45	1,51
6	Didi	46,33	46	2,86

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
7	Rendi	46,00	47	3,38
8	Bayu	48,90	48	0,04
9	Hanifar	38,20	49	7,95
10	Helmi	51,34	50	2,64
11	Anjiko	41,17	51	4,84
12	Burhan	50,62	57	2,57
13	Novem	45,81	46	2,73
14	Dani	42,94	45	1,6
15	Fajar	46,23	54	2,31
16	Edward	48,86	55	2,54
17	Restu	50,08	52	1,73



Gambar 4.5 Grafik perbandingan estimasi lebar badan dan lebar badan sebenarnya

Pengujian kali ini dilakukan pada 17 subjek. 12 subjek pertama merupakan subjek pelatihan, dan 5 lainnya merupakan subjek yang tidak

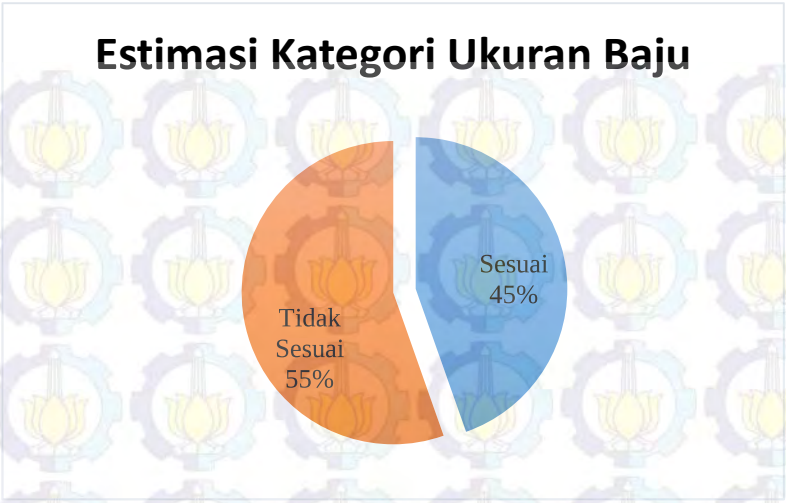
dilatihkan. Setelah melakukan pengambilan data, hasil estimasi lebar badan untuk pada 17 subjek pengujian memiliki nilai error absolut antara 0,04% hingga 7,95% dengan rata-rata nilai error absolut sebesar 2,68%. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 2.

4.2.3 Penentuan Kategori Baju

Penentuan kategori baju didapatkan dari hasil dari estimasi lebar badan yang dikonversikan kedalam kategori S, M, L, XL dan - berdasarkan pada tabel 3.1. Seperti yang terdapat pada gambar 4.7 hasil dari ukuran baju akan muncul setelah proses pengambilan data yang dilanjutkan dengan proses *feedforward* pada *Neural Networks*. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 3.

Tabel 4.4 Perbandingan estimasi ukuran baju dan ukuran sebenarnya

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Alka	-	-	Sesuai
2	David	-	-	Sesuai
3	Wahyudi	-	-	Sesuai
4	Fadli	S	-	Tidak sesuai
5	Firdaus	S	-	Tidak sesuai
6	Didi	S	S	Sesuai
7	Rendi	S	S	Sesuai
8	Bayu	S	S	Sesuai
9	Hanifar	-	S	Tidak sesuai
10	Helmi	M	M	Sesuai
11	Anjiko	-	M	Tidak sesuai
12	Burhan	M	XL	Tidak sesuai
13	Novem	-	S	Tidak sesuai
14	Dani	-	-	Sesuai
15	Fajar	S	L	Tidak sesuai
16	Edward	S	L	Tidak sesuai
17	Restu	M	M	Sesuai



Gambar 4.6 Grafik estimasi kategori ukuran baju

Gambar 4.7 Hasil dari estimasi lebar badan dan kategori ukuran pakaian

Berdasarkan hasil pengujian yang dilakukan pada 17 subjek, terdapat 3 estimasi ukuran baju yang tidak sesuai dengan ukuran baju yang sebenarnya, dikarenakan adanya pergeseran nilai estimasi lebar badan dengan lebar badan yang sebenarnya.

4.3 Pelatihan Data Pada Jarak 1200, 1500, dan 1700 Milimeter

Pengujian ini dilakukan untuk membandingkan keluaran sistem antara pelatihan data pada jarak 1500 mm dan pelatihan data pada jarak 1200 mm, 1500, serta 1800 mm. Data yang dilatihkan didapatkan dari pengambilan data pada 12 subjek. Setiap subjek akan diambil data sebanyak lima kali pada jarak 1200mm, 1500mm dan 1700mm. Sehingga setiap subjek akan memiliki 15 data untuk dilatihkan.

4.3.1 Pengukuran Jarak Antar Bahu

Terlihat pada gambar 4.8 bahwa subjek berada diluar jarak yang dianjurkan yaitu 1500 mm dan gambar yang ditankap Kinect menjadi merah sebagai indikatornya. Dapat dilihat pada tabel 4.5 estimasi jarak antar bahu pada jarak 1200 mm cukup akurat dengan rata-rata error absolut keseluruhan 6,39%. Begitu juga pada tabel 4.6 estimasi jarak antar bahu pada jarak 1500 mm dengan rata-rata error absolut keseluruhan 7,78%. Hal yang sama juga terdapat pada tabel 4.7 estimasi jarak antar bahu pada jarak 1500 mm memiliki rata-rata error absolut keseluruhan sebesar 7,46%. Untuk penentuan estimasi jarak antar bahu dapat dilakukan pada jarak 1200 hingga 1700 mm. Hasil pengujian secara lengkap dapat dilihat pada lampiran tabel 4, tabel 5 dan tabel 6.

Gambar 4.8 Hasil estimasi jarak antar bahu diluar jarak yang dianjurkan

Tabel 4.5 Hasil estimasi jarak antar bahu pada jarak 1200 mm

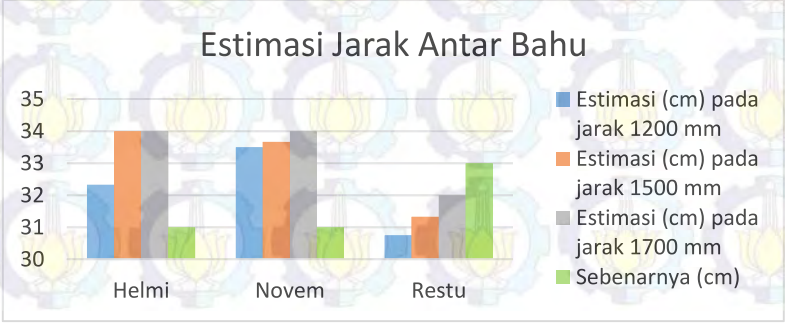
No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	32,33	31	4,30
2	Novem	33,50	31	8,06
3	Restu	30,75	33	6,82

Tabel 4.6 Hasil estimasi jarak antar bahu pada jarak 1500 mm

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	34,00	31	9,68
2	Novem	33,67	31	8,60
3	Restu	31,33	33	5,05

Tabel 4.7 Hasil estimasi jarak antar bahu pada jarak 1700 mm

No	Nama	Jarak Antar Bahu		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	34,00	31	9,68
2	Novem	34,00	31	9,68
3	Restu	32,00	33	3,03



Gambar 4.9 Grafik estimasi jarak antar bahu diluar jarak yang dianjurkan

4.3.2 Estimasi Lebar Badan

Pengujian kali ini juga dilakuan pada dua jarak yang berbeda yaitu pada jarak 1200 mm, 1500 mm, dan 1700 mm. Terlihat pada tabel 4.8 yaitu pada jarak 1200 mm estimasi lebar badan yang didapatkan keseluruhan rata-rata error absolut sebesar 18,85%. Pada tabel 4.9, pada jarak 1500 mm didapatkan keseluruhan error rata-rata absolut sebesar 46,36%. Sedangkan pada tabel 4.10, pada jarak 1700 mm didapatkan keseluruhan error rata-rata absolut sebesar 60,6%. Hasil pengujian pada jarak 1200 mm hingga 1700 mm secara lengkap terdapat pada lampiran tabel 7, 8, dan 9. Berdasarkan lampiran tabel 7, 8, dan 9 estimasi lebar badan pada jarak 1200 mm hingga 1700 mm cenderung tidak stabil dan tidak akurat. Hal tersebut dikarekan error pada proses pelatihan *neural networks* masih besar dan tidak bisa konvergen.

Gambar 4.10 Hasil estimasi lebar badan diluar jarak yang dianjurkan

Tabel 4.8 Hasil estimasi lebar badan pada jarak 1200 mm

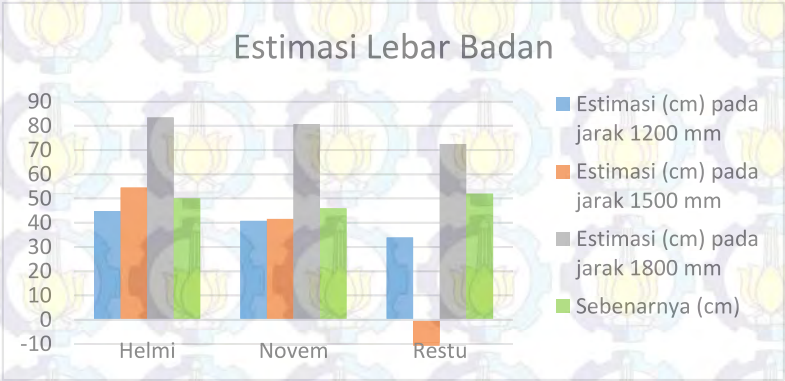
No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	44,74	50	10,52
2	Novem	40,77	46	11,38
3	Restu	33,98	52	34,66

Tabel 4.9 Hasil estimasi lebar badan pada jarak 1500 mm

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	54,51	50	9,02
2	Novem	41,62	46	9,53
3	Restu	-10,68	52	120,54

Tabel 4.10 Hasil estimasi lebar badan pada jarak 1700 mm

No	Nama	Lebar Badan		Error Absolut (%)
		Rata - rata Estimasi (cm)	Sebenarnya (cm)	
1	Helmi	83,45	50	66,90
2	Novem	80,71	46	75,45
3	Restu	72,51	52	39,44



Gambar 4.11 Grafik estimasi lebar badan diluar jarak yang dianjurkan

4.3.3 Penentuan Kategori Baju

Berdasarkan estimasi lebar badan yang didapatkan pada tabel 4.10, tabel 4.11, dan juga tabel 4.12 maka didapatkan hasil kategori ukuran baju pada jarak 1200 mm dan juga 1800 mm adalah seperti berikut ini.

Tabel 4.11 Hasil estimasi ukuran baju pada jarak 1200 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	-	M	Tidak sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

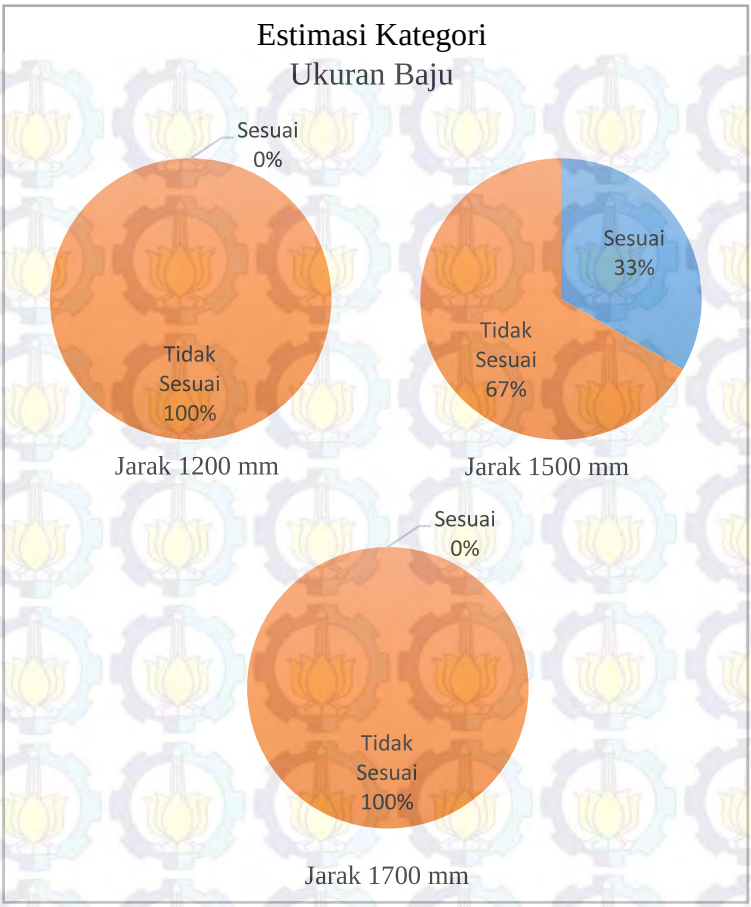
Tabel 4.12 Hasil estimasi ukuran baju pada jarak 1500 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	M	M	Sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

Tabel 4.13 Hasil estimasi ukuran baju pada jarak 1700 mm

No	Nama	Rata - rata Estimasi Ukuran Baju	Ukuran Baju Sebenarnya	Keterangan
1	Helmi	-	M	Tidak sesuai
2	Novem	-	S	Tidak sesuai
3	Restu	-	M	Tidak sesuai

Pada gambar 4.12 presentase ketidak sesuaian estimasi ukuran baju dengan ukuran baju sebenarnya mencapai 100% pada jarak 1200 mm dan 1700 mm. Sedangkan pada jarak 1500 mm presentase ketidak sesuaiannya sebesar 67%. Hal ini dikarenakan nilai estimasi lebar badan yang bergeser sangat jauh dari lebar badan sebenarnya. Hasil pengujian secara lengkap untuk proses penentuan estimasi kategori ukuran baju dapatt dilihat pada lampiran tabel 10, lampiran tabel 11, dan lampiran tabel 12.



Gambar 4.12 Grafik estimasi kategori ukuran baju diluar jarak yang dianjurkan

BAB V
KESIMPULAN DAN SARAN

Berdasarkan hasil dari uji coba sistem terhadap beberapa subjek, maka dapat ditarik beberapa kesimpulan dan saran untuk pengembangan sistem kedepan.

5.1 Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan uji coba pada sistem yaitu:

1. Pengukuran jarak yang dapat dilakukan menggunakan kamera Kinect secara optimal yaitu antara 1000mm hingga 4000mm dengan rata-rata error absolut 0,59%.
2. Hasil dari pengukuran jarak antar bahu dengan metode jarak Euclidean dapat dilakukan pada jarak antara 1200 mm hingga 1800 mm. Pada jarak 1200 mm, 1500 mm, dan 1700 mm hasil pengukuran mengalami pergeseran yang cukup kecil ditunjukkan dengan nilai rata-rata error absolut 6,39%, 7,78% dan 7,46%.
3. Pada *neural networks* yang dilatihkan pada jarak 1500 mm, sistem dapat melakukan estimasi lebar badan cukup akurat dengan nilai rata-rata keseluruhan error absolut sebesar 7,25%.
4. Pada *neural networks* yang dilatihkan pada jarak 1200 mm, 1500 mm, dan 1700 mm terdapat hasil estimasi lebar badan yang tidak akurat. Hal tersebut ditunjukkan dengan nilai rata-rata error keseluruhan estimasi lebar badan sebesar 18,85%, 46,36% dan 7,46% pada masing-masing jarak.
5. Jarak antara pengguna dengan Kinect memberikan pengaruh besar pada proses estimasi lebar badan. Pelatihan data pada jarak 1500 mm memberikan hasil yang paling baik pada sistem ini.
6. Sistem mampu memberikan saran ukuran baju berdasarkan lebar badan yang didapatkan. Tetapi hasil ukuran baju yang disarankan dapat mengalami pergeseran. Pergeseran tersebut disebabkan karena adanya pergeseran nilai estimasi lebar badan dengan lebar badan yang sebenarnya.

5.2 Saran

Untuk mengembangkan tugas akhir ini penulis memiliki beberapa saran sebagai berikut:

1. Untuk meningkatkan presisi dari estimasi lebar badan, diperlukan penambahan set data uji dengan subjek yang beragam.
2. Metode estimasi lebar badan dengan menggunakan *Neural Networks* perlu diteliti lebih lanjut.
3. Diharapkan kerja sistem tidak terganggu bila ada orang yang berdiri disekitar pengguna.
4. Melakukan pengembangan untuk sistem pengukuran secara *online*

LAMPIRAN

Tabel 1 Perbandingan estimasi jarak antar bahu dan jarak antar bahu sebenarnya

No	Nama	Estimasi Jarak Antar Bahu (cm)												Rata- rata	Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12			
1	Alka	30	30	30	30	30	30	31	30	30	30	29	30	30,00	30	0,00
2	David	30	30	30	30	31	30	30	30	31	30	30	30	30,17	31	2,69
3	Wahyudi	31	31	31	31	30	31	31	30	30	31	31	31	30,75	29	6,03
4	Fadli	32	32	32	32	32	32	31	31	32	32	32	32	31,83	31	2,69
5	Firdaus	33	32	33	33	32	32	32	32	32	32	31	32	32,17	32	0,52
6	Didi	32	32	32	32	32	32	32	32	31	32	32	31	31,83	30	6,11
7	Rendi	33	33	33	33	32	33	33	33	32	33	33	33	32,83	31	5,91
8	Bayu	35	34	34	34	35	35	35	35	35	35	35	35	34,75	34	2,21
9	Hanifar	33	33	33	33	33	33	32	32	33	33	33	33	32,83	32	2,60
10	Helmi	33	31	33	33	33	33	33	33	33	33	33	31	32,67	31	5,38
11	Anjiko	33	33	33	33	33	33	33	33	33	34	33	33	33,08	33	0,25
12	Burhan	36	36	36	36	36	36	36	35	36	36	36	36	35,92	35	2,62
13	Novem	34	34	34	34	34	34	34	34	34	33	34	32	33,75	30	12,50
14	Dani	34	31	31	34	34	34	34	34	34	34	34	34	33,50	31	8,06
15	Fajar	35	34	35	34	35	35	34	35	35	35	35	35	34,75	32	8,59
16	Edward	36	36	36	36	35	36	36	34	36	36	36	36	35,75	35	2,14
17	Restu	31	32	31	30	30	31	31	31	31	31	31	31	30,92	33	6,31

Tabel 2 Perbandingan estimasi lebar badan dan lebar badan sebenarnya

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Alka	44,68	49,17	47,27	47,94	44	36,98	42,27	35,24	36,97	38,19	35,77	37,79	41,36	39	6,04
2	David	37,12	38,18	37,87	41,81	40,21	40,67	38	38,48	40,29	38,26	39,49	39,28	39,14	40	2,15
3	Wahyudi	41,26	47,68	42,04	39,58	39,58	37,87	39,37	46,62	47,23	43,67	44,56	45,9	42,95	42	2,25
4	Fadli	49,57	48,94	54,91	55,35	51,89	44,81	46,29	47,21	48,8	45,5	45,91	45,41	48,72	44	10,72
5	Firdaus	42,05	48,79	49,01	44,32	51,56	50,95	49,25	49,34	49,56	49,84	48,9	49,56	48,59	45	7,99
6	Didi	46,88	46,58	47,32	49,19	47,57	47,87	45,54	44,77	44,22	47,05	44,47	44,45	46,33	46	0,71
7	Rendi	47,61	47,75	48,09	47,58	46,09	48,53	48,01	48,59	46,4	47,79	46,43	46,63	46,00	47	2,13
8	Bayu	54,53	51,01	50,79	49,59	52,46	48,06	44,63	47,71	48,02	46,12	49,27	44,56	48,90	48	1,87
9	Hanifar	35,08	39,63	42,34	33,37	38,97	34,18	33,85	40,34	45,1	42,38	35,1	38,05	38,20	49	22,04
10	Helmi	41,75	50,71	58,7	54,41	55,2	55,39	51,32	47,02	52,9	49,3	47,15	52,19	51,34	50	2,67
11	Anjiko	41,4	42,55	39,9	47,05	45,61	37,35	40,07	48,53	38,69	40,14	34,46	38,28	41,17	51	19,28
12	Burhan	53,74	50,6	48,8	45,91	43,48	44,5	52,24	57,46	58,78	51,02	52,62	48,31	50,62	57	11,19
13	Novem	45,93	47,26	44,79	44	44,85	45,12	46,36	48,7	45,49	44,63	44,65	47,93	45,81	46	0,41
14	Dani	42,63	38,27	45,72	39,1	47,8	43,37	49,4	38,98	40,37	43,85	40,08	45,73	42,94	45	4,57
15	Fajar	31,28	30,56	47,01	50,23	55,25	53,2	53,56	44,41	51,57	38,14	50,67	48,82	46,23	54	14,40
16	Edward	48,94	46,97	45,54	49,77	53,6	42,15	51,6	46,92	48,02	49,96	53,117	49,75	48,86	55	11,16
17	Restu	48,75	50,06	50,01	51,42	49,49	50,33	49,99	48,9	52	49,94	51	49,05	50,08	52	3,70

Tabel 3 Perbandingan estimasi ukuran baju dan ukuran sebenarnya

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Alka	-	S	S	S	-	-	-	-	-	-	-	-	-	75,00
2	David	-	-	-	-	-	-	-	-	-	-	-	-	-	100,00
3	Wahyudi	-	S	-	-	-	-	-	S	S	-	-	-	-	75,00
4	Fadli	S	S	L	L	M	-	S	S	S	-	-	-	-	41,67
5	Firdaus	-	S	S	-	M	M	S	S	S	S	S	S	-	16,67
6	Didi	S	S	S	S	S	S	-	-	-	S	-	-	S	58,33
7	Rendi	S	S	S	S	S	S	S	S	S	S	S	S	S	100,00
8	Bayu	L	M	M	S	M	S	-	S	S	S	S	-	S	50,00
9	Hanifar	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
10	Helmi	-	M	XL	L	L	L	M	S	M	S	S	M	M	33,33
11	Anjiko	S	-	-	-	S	-	-	-	S	-	-	-	M	0,00
12	Burhan	M	M	S	-	-	-	M	XL	XL	M	M	S	XL	25,00
13	Novem	-	S	-	-	-	-	S	S	-	-	-	S	S	41,67
14	Dani	-	-	-	-	S	-	S	-	-	-	-	-	-	83,33
15	Fajar	-	-	S	M	L	M	M	-	M	-	M	S	L	8,33
16	Edward	S	S	-	S	M	-	M	S	S	S	M	S	L	0,00
17	Restu	S	M	M	M	S	M	S	S	M	S	M	S	M	50,00

Tabel 4 Hasil estimasi jarak antar bahu pada jarak 1200 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	33	33	33	31	33	33	33	31	31	33	33	31	32,33	31	4,30
2	Novem	34	34	34	33	33	34	33	34	34	33	33	33	33,50	31	8,06
3	Restu	31	31	30	31	31	31	31	31	31	30	31	30	30,75	33	6,82

Tabel 5 Hasil estimasi jarak antar bahu pada jarak 1500 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
2	Novem	34	33	34	33	34	33	34	34	34	34	33	34	33,67	31	8,60
3	Restu	31	31	31	31	31	32	32	31	32	31	31	32	31,33	33	5,05

Tabel 6 Hasil estimasi jarak antar bahu pada jarak 1700 mm

No	Nama	Estimasi Jarak Antar Bahu (cm)													Jarak Antar Bahu Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
2	Novem	34	34	34	34	34	34	34	34	34	34	34	34	34,00	31	9,68
3	Restu	32	32	32	32	32	32	32	32	32	32	32	32	32,00	33	3,03

Tabel 7 Hasil estimasi lebar badan pada jarak 1200 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	41,1	45.86	36.64	47.08	45,78	50,37	48,1	44.04	45.55	48,38	43,62	35,83	44,74	50	10,52
2	Novem	38,9	41,56	41,58	39,87	40,88	41,14	40,12	41,58	40,5	41,05	40,7	41,33	40,77	46	11,38
3	Restu	45,66	26,66	38,18	36,48	36,48	18,05	31,94	27,2	38,45	30,25	36,48	41,89	33,98	52	34,66

Tabel 8 Hasil estimasi lebar badan pada jarak 1500 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	51,41	48,33	56,02	58,18	55,65	56,8	56,08	57,44	57,44	58,18	48,87	49,73	54,51	50	9,02
2	Novem	38,74	38,5	37,18	37,97	46,45	40,36	39,58	51,08	41,01	39,58	48,57	40,36	41,62	46	9,53
3	Restu	21,06	-12,6	4,06	-16	-16	-3,1	-16	-3,8	-22	-23,2	-31	-8,9	-10,6	52	120,43

Tabel 9 Hasil estimasi lebar badan pada jarak 1700 mm

No	Nama	Estimasi Lebar Badan (cm)													Lebar Badan Sebenarnya (cm)	Rata-rata Error absolut (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Rata- rata		
1	Helmi	86,29	85,75	80,21	86,29	85,75	84,65	84,65	76,88	86,71	79,14	78,77	86,29	83,45	50	66,90
2	Novem	84,94	86,54	82,33	77,44	80,01	84,55	77,23	76,05	83,42	76,81	84,35	74,82	80,71	46	75,45
3	Restu	88,86	78,63	60,98	75,19	70,91	75,19	79,36	77,96	78,63	61,74	75,19	47,48	72,51	52	39,44

Tabel 10 Hasil estimasi ukuran baju pada jarak 1200 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	S	-	M	S	-	-	S	-	-	M	8,33
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00

Tabel 11 Hasil estimasi ukuran baju pada jarak 1500 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	XL	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00

Tabel 12 Hasil estimasi ukuran baju pada jarak 1700 mm

No	Nama	Estimasi Ukuran Baju												Ukuran Baju Sebenarnya	Presentase Kesesuaian Estimasi Ukuran Baju dengaan Ukuran Baju Sebenarnya (%)
		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12		
1	Helmi	-	-	-	-	-	-	-	-	-	-	-	-	M	0,00
2	Novem	-	-	-	-	-	-	-	-	-	-	-	-	S	0,00
3	Restu	-	-	-	-	-	-	-	-	-	-	S	-	M	0,00

Listing program Visual Studio C++

```
double round(double d)
{
  return floor(d + 0.5);
}
void ambil_in(int jenis_out)
{
   DataIn[jenis_out][0]=lebar_p;
   DataIn[jenis_out][1]=lebar_s;
   DataIn[jenis_out][2]=LCdepth;
   DataIn[jenis_out][3]=RCdepth;
}
void save_trainsets(int jumlahdata, int index)
{
   ofstream outFile;
   string a ("Data (");
   string c (").txt");
   string nama_data;
   stringstream b;
   stringstream d;
   b<<jumlahdata;
   d<<index;
   nama_data= d.str()+a+b.str()+c;
   outFile.open(nama_data);
   for (int i = 0 ; i < input; i++)
   {
        outFile << DataIn[jumlahdata-1][i]<<endl;
        mbuh =jumlahdata;
   }
   for (int i = 0 ; i < Output; i++)
   {
        outFile << OutDesire[jumlahdata-1][i]<<endl;
   }
   outFile.close();
   if (jumlahdata==20)
   {
        jumlahdata++;
        stringstream b;
        b<<jumlahdata;
        nama_data=d.str()+a+b.str()+c;
        outFile.open(nama_data);
```

```
        for (int k=0; k<jumlahdata-1; k++)
        {
            for (int i = 0 ; i < input; i++)
            {
                outFile << DataIn[k][i]<<endl;
            }
            for (int i = 0 ; i < Output; i++)
            {
                outFile << OutDesire[k][i]<<endl;
            }
        }
        outFile.close();
    }
}
void ambil_out()
{
    RandomIn[0]=lebar_s;
    RandomIn[1]=LSdepth;
    RandomIn[2]=RSdepth;
    RandomIn[3]=LSdepth - chestdepth;
    RandomIn[4]=LSdepth - bellydepth;
}
void IO_txt()
{
    ifstream inFile;
    inFile.open("Input.txt");
    for (int i = 0; i<JenisOut; i++)
    {
        for (int j = 0 ; j < input; j++)
        {
        inFile >> DataIn[i][j];
        }
    }
    inFile.close();
    inFile;
    inFile.open("Output.txt");
    for (int i = 0; i<JenisOut; i++)
    {
        for (int j = 0 ; j < Output; j++)
        {
        inFile >> OutDesire[i][j];
        }
    }
```

```
    inFile.close();
}
void denormalize ()
{
    double xmin = OutDesire[0][0];
    double xmax = OutDesire[0][0];
    for(int i=0;i<JenisOut;i++)
    {
        if(xmin>OutDesire[i][0])
        {
            xmin=OutDesire[i][0];
        }
        else if(xmax<OutDesire[i][0])
        {
            xmax = OutDesire[i][0];
        }
    }
size_est = 0.5*(xo3[0]+1)*(xmax-xmin) + xmin;
}
void normalize(int pola, int jenis_out, int trainOrRandom)
{
    double xmin = DataIn[pola][pola];
    double xmax = DataIn[pola][pola];
    if(trainOrRandom==1)
    {
        for(int i=0;i<jenis_out;i++)
        {
            if(xmin>DataIn[i][pola])xmin=DataIn[i][pola];
            else if(xmax<DataIn[i][pola])
                xmax = DataIn[i][pola];
RandomIn[pola]= 2* (RandomIn[pola]-xmin) / (xmax-xmin) - 1;
    }
}

void feedforward()
{
    //----------------- input --> H1
    for (int i = 0; i < HiddenLayer1; i++)
    {
        v1[i]=bias1[i];
        for (int j = 0; j < input; j++)
        {
            v1[i] = v1[i] + (W1[i][j]*RandomIn[j]);
```

```
        }
        xo1[i] = 2 / (1+exp(-2*v1[i])) -1;
    }
    //----------------- input --> H2
    for (int i = 0; i < Output; i++)
    {
        v2[i]=bias2[i];
        for (int j = 0; j < HiddenLayer1; j++)
        {
            v2[i] = v2[i] + (W2[i][j]*xo1[j]);
        }
        xo3[i] = v2[i];
    }
}
void load_w1()
{
    ifstream inFile;
    inFile.open("w1.txt");
    for (int i = 0; i<HiddenLayer1; i++)
    {
        for (int j = 0 ; j < input; j++)
        {
        inFile >> W1[i][j];
        }
    }
    inFile.close();
}
void load_w2()
{
    ifstream inFile;
    inFile.open("w2.txt");
    for (int i = 0; i<Output; i++)
    {
        for (int j = 0 ; j < HiddenLayer1; j++)
        {
        inFile >> W2[i][j];
        }
    }
    inFile.close();
}
void load_bias1()
{
    ifstream inFile;
```

```
    inFile.open("bias1.txt");
    for (int i = 0; i<HiddenLayer1; i++)
    {
        inFile >> bias1[i];
    }
    inFile.close();
}
void load_bias2()
{
    ifstream inFile;
    inFile.open("bias2.txt");
    for (int i = 0; i<Output; i++)
    {
        inFile >> bias2[i];
    }
    inFile.close();
}
void rata2()
{
    if (kondisi <rata)
    {
        hitung[kondisi][0]=lebar_s;
        hitung[kondisi][1]=LSdepth;
        hitung[kondisi][2]=RSdepth;
        hitung[kondisi][3]=LSdepth - chestdepth;
        hitung[kondisi][4]=LSdepth - bellydepth;
    }
}
void load_train()
{
    load_w1();
    load_w2();
    load_bias1();
    load_bias2();
}
void NN_main()
{
    normalize(0,JenisOut,1);
    normalize(1,JenisOut,1);
    normalize(2,JenisOut,1);
    normalize(3,JenisOut,1);
    normalize(4,JenisOut,1);
    feedforward();
```

```
    denormalize();
}
void average()
{
    if (kondisi==rata)
    {

        for(int i=0; i<rata;i++)
        {
            RandomIn[0] = hitung[i][0];
            RandomIn[1] = hitung[i][1];
            RandomIn[2] = hitung[i][2];
            RandomIn[3] = hitung[i][3];
            RandomIn[4] = hitung[i][4];

            NN_main();
            temp[i]=size_est;
            size_avg+=temp[i];
        }
        size_avg= size_avg/rata;
        int c;
        c=size_avg*100;
        size_avg=c/100.;
        kondisi++;
    }

}
void kosongkan()
{
    for (int i=0; i<rata;i++)
    {
        hitung[i][0]=0;
        hitung[i][1]=0;
        hitung[i][2]=0;
        hitung[i][3]=0;
        hitung[i][4]=0;
        size_avg=0;
        temp[i]=0;
        kondisi=0;
        label4->Text=String::Concat("...");
        label20->Text=String::Concat("...");
        label21->Text=String::Concat("...");
    }
```

```
}
void kategori ()
{
   if ((size_avg<46) || (size_avg>60))
        {
            label4->Text= "-";
        }
   if ((size_avg<50) && (size_avg>46))
        {
            label4->Text= "S";
        }
   if ((size_avg<54) && (size_avg>50))
        {
            label4->Text= "M";
        }
   if ((size_avg<56) && (size_avg>54))
        {
            label4->Text= "L";
        }
   if ((size_avg<60) && (size_avg>56))
        {
            label4->Text= "XL";
        }
   kondisi=1000;
}
HRESULT inisialisasi_depth()
{
//====================   inisialisasi penggunaan depth
   hr= m_pNuiSensor-
>NuiInitialize(NUI_INITIALIZE_FLAG_USES_COLOR|NUI_INITIALIZ
E_FLAG_USES_DEPTH | NUI_INITIALIZE_FLAG_USES_SKELETON);
   if(SUCCEEDED(hr))
   {
//====================   kalo berhasil inisialisasi, di
create event
        m_hNextColorFrameEvent = CreateEvent(NULL, TRUE,
FALSE, NULL);

   m_hNextDepthFrameEvent=CreateEvent(NULL,TRUE,FALSE,NULL)
;
        m_hNextSkeletonEvent = CreateEvent(NULL, TRUE,
FALSE, NULL);
```

```
        hr = m_pNuiSensor-
>NuiImageStreamOpen(NUI_IMAGE_TYPE_COLOR,NUI_IMAGE_RESOLUTI
ON_640x480, 0, 2, m_hNextColorFrameEvent,
&m_pColorStreamHandle);
        if( FAILED( hr ) )return hr;
        }
//====================    buka depth stream untuk dapat
frame depth
        hr = m_pNuiSensor-
>NuiImageStreamOpen(NUI_IMAGE_TYPE_DEPTH,
NUI_IMAGE_RESOLUTION_640x480,0,2,m_hNextDepthFrameEvent,&m_
pDepthStreamHandle);
        if( FAILED( hr ) )return hr;
        hr = m_pNuiSensor-
>NuiSkeletonTrackingEnable(m_hNextSkeletonEvent,NUI_SKELETO
N_TRACKING_FLAG_ENABLE_IN_NEAR_RANGE);
        if( FAILED( hr ) )return hr;
    return hr;
}
//====================    cek sensor
HRESULT initial()
{
    iSensorCount=0;
    hr = NuiGetSensorCount(&iSensorCount);
    if(FAILED(hr))return hr;
    for (int i = 0 ; i < iSensorCount; ++i)
    {
//====================    buat index status sensor untuk di
cek statusnya, kalo gak bisa lanjut aja
        hr = NuiCreateSensorByIndex ( i, &pNuiSensor);
        if(FAILED(hr))continue;
//====================    cek statusnya, kalo OK di
inisialise
        hr = pNuiSensor-> NuiStatus();
        if(S_OK==hr)
        {
            m_pNuiSensor=pNuiSensor;
            break;
        }
//====================    kalo gak ok, release aja, gak
kepake kok
        pNuiSensor->Release();
    }
```

```
//==================    kalo ok berarti pNuiSensornya !=
NULL
   if(NULL!= m_pNuiSensor)hr= inisialisasi_depth();
//==================    kalo gak ok nuisensor 2=null
   if(NULL==m_pNuiSensor||FAILED (hr))return E_FAIL;
   else
   {
//==================    siapin frame windows penamung img
depth
       color = cvCreateImageHeader(cvSize(cColorWidth,
cColorHeight), IPL_DEPTH_8U, 4);

   depthh=cvCreateImageHeader(cvSize(cDepthWidth,cDepthHeig
ht),IPL_DEPTH_8U, cBytesPerPixel);
       color_r = cvCreateImage(cvGetSize(color), color-
>depth, 1);
       color_g = cvCreateImage(cvGetSize(color), color-
>depth, 1);
       color_b = cvCreateImage(cvGetSize(color), color-
>depth, 1);
       m_depthRGBX= new
BYTE[cDepthWidth*cDepthHeight*cBytesPerPixel];
       //cvNamedWindow("Color Image", CV_WINDOW_AUTOSIZE);
       cvNamedWindow("Depth Image",CV_WINDOW_AUTOSIZE);
   }
}
void PointSkel ( IplImage* img, Vector4 posisiskel )
{
   FLOAT coordepthX = 0, coordepthY = 0;
   NuiTransformSkeletonToDepthImage(posisiskel,
&coordepthX, &coordepthY, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX = 0, coorrgbY = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX,
(LONG)coordepthY, 0, &coorrgbX, &coorrgbY );
   cvCircle(img,cvPoint(coorrgbX,coorrgbY),10,CV_RGB(0,255,
0),5,8,0);
}
void DrawLine( IplImage* img, Vector4 joint0, Vector4
joint1)
```

```
{
   FLOAT coordepthX0 = 0, coordepthY0 = 0;
   NuiTransformSkeletonToDepthImage(joint0, &coordepthX0,
&coordepthY0, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX0 = 0, coorrgbY0 = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX0,
(LONG)coordepthY0, 0, &coorrgbX0, &coorrgbY0 );
   FLOAT coordepthX1 = 0, coorcoordepthY0 = 0;
   NuiTransformSkeletonToDepthImage(joint1, &coordepthX1,
&coorcoordepthY0, NUI_IMAGE_RESOLUTION_640x480);
   LONG coorrgbX1 = 0, coorrgbY1 = 0;
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480, 0, (LONG)coordepthX1,
(LONG)coorcoordepthY0, 0, &coorrgbX1, &coorrgbY1 );
   // gambar lingkaran untuk setiap joint dan bikin
sambungan tulang
   cvLine(img, cvPoint(coorrgbX0, coorrgbY0),
cvPoint(coorrgbX1, coorrgbY1), CV_RGB(50,255,50),3,8,0);
   cvCircle(img, cvPoint(coorrgbX0, coorrgbY0),
5,CV_RGB(255,0,0),3,8,0);
}
void DrawBone(
   IplImage* img,
    const NUI_SKELETON_DATA & skel,
    NUI_SKELETON_POSITION_INDEX joint0,
    NUI_SKELETON_POSITION_INDEX joint1)
{
    NUI_SKELETON_POSITION_TRACKING_STATE joint0State =
skel.eSkeletonPositionTrackingState[joint0];
    NUI_SKELETON_POSITION_TRACKING_STATE joint1State =
skel.eSkeletonPositionTrackingState[joint1];
   //If we can't find either of these joints, exit
    if (joint0State == NUI_SKELETON_POSITION_NOT_TRACKED ||
joint1State == NUI_SKELETON_POSITION_NOT_TRACKED)
    {
        return;
    }
```

```
    const Vector4 joint0Position =
skel.SkeletonPositions[joint0];
    const Vector4 joint1Position =
skel.SkeletonPositions[joint1];
   sendi[joint0] = joint0Position;
   NuiTransformSkeletonToDepthImage(sendi[joint0],
&SenDepthX[joint0], &SenDepthY[joint0],
NUI_IMAGE_RESOLUTION_640x480);
   m_pNuiSensor-
>NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution
(NUI_IMAGE_RESOLUTION_640x480,
       NUI_IMAGE_RESOLUTION_640x480,
0,(LONG)SenDepthX[joint0], (LONG)SenDepthY[joint0], 0,
&SenPosX[joint0], &SenPosY[joint0] );
    // Don't draw if both points are inferred
    if (joint0State == NUI_SKELETON_POSITION_INFERRED ||
joint1State == NUI_SKELETON_POSITION_INFERRED)
       DrawLine(img, joint0Position, joint1Position);
    // We assume all drawn bones are inferred unless BOTH
joints are tracked
    if (joint0State == NUI_SKELETON_POSITION_TRACKED &&
joint1State == NUI_SKELETON_POSITION_TRACKED)
       DrawLine(img, joint0Position, joint1Position);
}

void DrawSkeleton(IplImage* img, const NUI_SKELETON_DATA&
skel)
{
    // Render Torso
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_CENTER);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_LEFT);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SHOULDER_RIGHT);
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_CENTER,
NUI_SKELETON_POSITION_SPINE);
    DrawBone(img, skel, NUI_SKELETON_POSITION_SPINE,
NUI_SKELETON_POSITION_HIP_CENTER);
```

```
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_CENTER,
NUI_SKELETON_POSITION_HIP_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_CENTER,
NUI_SKELETON_POSITION_HIP_RIGHT);
    // Left Arm
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_LEFT,
NUI_SKELETON_POSITION_ELBOW_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ELBOW_LEFT,
NUI_SKELETON_POSITION_WRIST_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_WRIST_LEFT,
NUI_SKELETON_POSITION_HAND_LEFT);
    // Right Arm
    DrawBone(img, skel,
NUI_SKELETON_POSITION_SHOULDER_RIGHT,
NUI_SKELETON_POSITION_ELBOW_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ELBOW_RIGHT,
NUI_SKELETON_POSITION_WRIST_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_WRIST_RIGHT,
NUI_SKELETON_POSITION_HAND_RIGHT);
    // Left Leg
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_LEFT,
NUI_SKELETON_POSITION_KNEE_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_KNEE_LEFT,
NUI_SKELETON_POSITION_ANKLE_LEFT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ANKLE_LEFT,
NUI_SKELETON_POSITION_FOOT_LEFT);

    // Right Leg
    DrawBone(img, skel, NUI_SKELETON_POSITION_HIP_RIGHT,
NUI_SKELETON_POSITION_KNEE_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_KNEE_RIGHT,
NUI_SKELETON_POSITION_ANKLE_RIGHT);
    DrawBone(img, skel, NUI_SKELETON_POSITION_ANKLE_RIGHT,
NUI_SKELETON_POSITION_FOOT_RIGHT);
   LS = skel.SkeletonPositions[4];
   RS= skel.SkeletonPositions[8];
   chest = skel.SkeletonPositions[2];
   wrist= skel.SkeletonPositions[6];
   belly = skel.SkeletonPositions[1];
}

void ProcessSkel(IplImage* img)
```

```
{
       for (int i = 0; i < NUI_SKELETON_COUNT; i++)
    {
        const NUI_SKELETON_DATA & skel =
skeletonFrame.SkeletonData[i];
       switch (skel.eTrackingState)
        {
       case NUI_SKELETON_TRACKED:
           orang=1;
            DrawSkeleton(img, skel);
           cvCircle(depthh, cvPoint(SenPosX[4],
SenPosY[2]), 3,CV_RGB(0,0,255),6,0); //LS
           cvCircle(depthh, cvPoint(SenPosX[8],
SenPosY[2]), 3,CV_RGB(0,0,255),6,0); //RS
           cvCircle(depthh, cvPoint(SenPosX[2],
(SenPosY[4]+SenPosY[8])/2),3,CV_RGB(255,255,255),6,0);
//chest
           cvCircle(depthh, cvPoint(SenPosX[1],
SenPosY[1]), 3,CV_RGB(0,0,255),6,0); //belly
            break;
        case NUI_SKELETON_POSITION_ONLY:
            PointSkel(img, skel.Position);
            break;
       default:
           orang++;
        }
    }
}
void get_collor()
{
   if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextColorFrameEvent, 0))
   {
       hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pColorStreamHandle, 0,
&imageFrame);
       if (FAILED(hr))  return;
       INuiFrameTexture * pTexture =
imageFrame.pFrameTexture;
       NUI_LOCKED_RECT LockedRect;
       // Lock the frame data so the Kinect knows not to
modify it while we're reading it
       pTexture->LockRect(0, &LockedRect, NULL, 0);
```

```
        // Make sure we've received valid data
        if (LockedRect.Pitch != 0)
        {
            BYTE * pBuffer = (BYTE*) LockedRect.pBits;
            cvSetData(color, pBuffer, LockedRect.Pitch);
            ProcessSkel(color);
   cvLine(color,cvPoint(210,0),cvPoint(210,480),CV_RGB(255,
0,0),2,0);
   cvLine(color,cvPoint(420,0),cvPoint(420,480),CV_RGB(255,
0,0),2,0);
            if (merah==1)
            {
                cvSplit(color,color_b, color_g, color_r,
NULL);
                cvSet(color_r, cvScalar(255), NULL);
            cvMerge(color_b,color_g,color_r,NULL,color);
            }
            pictureBox1->Image  = gcnew
System::Drawing::Bitmap(color->width,color->height,color-
>widthStep,System::Drawing::Imaging::PixelFormat::Format32b
ppRgb,(System::IntPtr) color->imageData);
            pictureBox1->Refresh();
        }
        // We're done with the texture so unlock it
        pTexture->UnlockRect(0);
        // Release the frame
        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pColorStreamHandle,
&imageFrame);
        cvWaitKey(1);
   }
}
void get_depth()
{
   int LSPos = SenPosX[4]+ SenPosY[2]*cDepthWidth;
   int RSPos = SenPosX[8]+ SenPosY[2]*cDepthWidth;
   int LCPos = SenPosX[4]+ SenPosY[4]*cDepthWidth;
   int RCPos = SenPosX[8]+ SenPosY[8]*cDepthWidth;
   int BellyPos = SenPosX[1]+ SenPosY[1]*cDepthWidth;
   int ChestPos = SenPosX[2]+
(SenPosY[4]+SenPosY[8])/2*cDepthWidth;
   int LSpx =0;
   int RSpx =0;
```

```
   int LCpx =0;
   int RCpx =0;
   int Chestpx =0;
   int Bellypx =0;
   if
(WAIT_OBJECT_0==WaitForSingleObject(m_hNextDepthFrameEvent,
0))
   {
       hr=m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pDepthStreamHandle,0,
&imageFrame);
       if(FAILED(hr))return;
       BOOL modenear = false;
       INuiFrameTexture* tekstur;
//====================   ambil informasi depth
       hr= m_pNuiSensor-
>NuiImageFrameGetDepthImagePixelFrameTexture(m_pDepthStream
Handle,&imageFrame,&modenear,&tekstur);
       if(FAILED(hr)) goto ReleaseFrame;
       NUI_LOCKED_RECT kuncirect;
//====================   kunci frame data saat pembacaan
       tekstur->LockRect(0,&kuncirect,NULL,0);
       if (kuncirect.Pitch!=0)
//====================   dapatin nilai min-max depth yang
reliable di current frame
       {
           int minDepth = (modenear ?
NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MINIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
           int maxDepth = (modenear ?
NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MAXIMUM) >> NUI_IMAGE_PLAYER_INDEX_SHIFT;
           BYTE* runwarna = m_depthRGBX;
           const NUI_DEPTH_IMAGE_PIXEL* BuffRC;
           const NUI_DEPTH_IMAGE_PIXEL* BuffLC;
           const NUI_DEPTH_IMAGE_PIXEL* BuffRS;
           const NUI_DEPTH_IMAGE_PIXEL* BuffLS;
           const NUI_DEPTH_IMAGE_PIXEL* BuffBelly;
           const NUI_DEPTH_IMAGE_PIXEL* BuffChest;
           const NUI_DEPTH_IMAGE_PIXEL* RunBuffer =
reinterpret_cast <const
NUI_DEPTH_IMAGE_PIXEL*>(kuncirect.pBits);
```

```
//====================    end pixelnya yaitu    start +
widht*height - 1
            const NUI_DEPTH_IMAGE_PIXEL* EndBuffer =
RunBuffer + (cDepthWidth*cDepthHeight) -1;
//====================    hitung distance depth (selisihnya)
            int selisih = 0;
            selisih = EndBuffer-RunBuffer;
            while (RunBuffer<EndBuffer)
            {
                USHORT depth= RunBuffer->depth;
                BYTE intensitasnya =
static_cast<byte>(depth >= minDepth && depth <= maxDepth ?
depth % 255:0);
                *(runwarna++) = intensitasnya;
                *(runwarna++) = intensitasnya;
                *(runwarna++) = intensitasnya;
                ++runwarna;
                ++RunBuffer;
            }
            RunBuffer = RunBuffer-selisih;

            int LS_s= LS.z*1000;
            int RS_s= RS.z*1000;
            int selisih_s = abs(LS_s - RS_s);
            if ( (LS_s >= 1000) && (selisih_s < 50))
            {
                FLOAT diff_x = LS.x - RS.x;
                FLOAT diff_y = LS.y - RS.y;
                FLOAT diff_z = LS.z - RS.z;
                FLOAT tinggi_x = chest.x - wrist.x;
                FLOAT tinggi_y = chest.y - wrist.y;
                FLOAT tinggi_z = chest.z - wrist.z;
                RS_s = RS.z;
                LS_s = LS.z;
                chest_s = chest.z;
                belly_s = belly.z;
                BuffLC = RunBuffer;
                BuffRC = RunBuffer;
                BuffChest = RunBuffer;
                BuffBelly = RunBuffer;
                BuffLS = RunBuffer;
                BuffRS = RunBuffer;
                LCpx = LCPos;
```

```
                RCpx = RCPos;
                Bellypx = BellyPos;
                Chestpx = ChestPos;
                LSpx = LSPos;
                RSpx = RSPos;


                BuffChest = BuffChest + Chestpx;
                BuffBelly = BuffBelly + Bellypx;
                BuffLS = BuffLS + LSpx;
                BuffRS = BuffRS + RSpx;
                BuffLC = BuffLC + LCpx;
                BuffRC = BuffRC + RCpx;
                bellydepth = BuffBelly->depth;
                chestdepth = BuffChest->depth;
                LSdepth = BuffLS->depth;
                RSdepth = BuffRS->depth;
                LCdepth = BuffLC->depth;
                RCdepth = BuffRC->depth;
                lebar_s =
round(sqrt(pow(diff_x,2)+pow(diff_y,2)+pow(diff_z,2))*100);
            }
            lebar_p = SenPosX[8] - SenPosX[4];
            USHORT * pBuff = (USHORT*)m_depthRGBX;
            cvSetData(depthh,pBuff,kuncirect.Pitch);
            if (orang <=50)
            {
   cvCircle(depthh,cvPoint(SenPosX[4],SenPosY[2]),3,CV_RGB(
0,0,255),6,0); //LS
   cvCircle(depthh,cvPoint(SenPosX[8],SenPosY[2]),3,CV_RGB(
0,0,255),6,0); //RS
   cvCircle(depthh,cvPoint(SenPosX[2],(SenPosY[4]+SenPosY[8
])/2),3,CV_RGB(0,0,255),6,0); //chest
   cvCircle(depthh,cvPoint(SenPosX[1],SenPosY[1]),3,CV_RGB(
0,0,255),6,0); //belly
            }
            if (orang >50)
            {
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//LS
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//RS
```

```
   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//chest

   cvCircle(depthh,cvPoint(0,0),3,CV_RGB(0,0,255),6,0);
//belly
            }
        }
        cvShowImage("Depth Image",depthh);
        tekstur->UnlockRect(0);
        tekstur->Release();
        ReleaseFrame:
        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pDepthStreamHandle,&imageFram
e);
        cvWaitKey(1);
    }
}

   private: wchar_t* string_to_wchartptr(System::String^
in){
                pin_ptr<const wchar_t> wch
=PtrToStringChars(in);
                size_t sizeInBytes = ((in->Length + 1) *
2);
                wchar_t* out =
(wchar_t*)calloc(sizeInBytes, sizeof(wchar_t));
                wcscpy_s(out, sizeInBytes, wch);
                return out;
            }
   private: bool KillProcessByName(wchar_t*
szProcessToKill){

                HANDLE hProcessSnap;
                HANDLE hProcess;
                PROCESSENTRY32 pe32;
                hProcessSnap =
CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0 );
                if(hProcessSnap == INVALID_HANDLE_VALUE){
                   return false;
                }

                pe32.dwSize = sizeof(PROCESSENTRY32);
                if(!Process32First(hProcessSnap, &pe32)){
```

```
                    CloseHandle(hProcessSnap);
                    return false;
                }
                do{
                    if(!wcscmp(pe32.szExeFile,
szProcessToKill)){
                        hProcess =
OpenProcess(PROCESS_TERMINATE, 0, pe32.th32ProcessID);
                        TerminateProcess(hProcess, 0);;
                        CloseHandle(hProcess);
                    }
                }while(Process32Next(hProcessSnap,
&pe32));
                CloseHandle(hProcessSnap);
                return true;
            }
    private: System::Void button1_Click(System::Object^
sender, System::EventArgs^  e) {
    mulai=1;
    button1->Enabled = false;
    int pos_bahu;
    LONG angle;
    IO_txt();
    load_train();
    // inisialisasi sensor kinect
    hr = initial();
    //NuiCameraElevationGetAngle(&angle);
    //if (angle!=0)
    //NuiCameraElevationSetAngle(0);
    while(true)
    {
        m_pNuiSensor->NuiSkeletonGetNextFrame( 0,
&skeletonFrame );
        get_collor();
        get_depth();
        jalan=1;
        jarak = (LSdepth+RSdepth)/2;
        label1->Text = String::Concat("",(jarak));
        pos_bahu= (abs (LSdepth-RSdepth));
        if (orang>= 700) orang=100;

        if(pos_bahu <=20 && (orang<=50))
            {
```

```
                lurus=1;
                label2->Text = String::Concat("LURUS");
            }
        else lurus =0;
        if (lurus ==0) label2->Text = String::Concat("TIDAK
LURUS");
//------------ambil data-----------------//
        if (orang <=50) //ini pengganti orang=1
        {merah=1;
            if((jarak>=1480) && (jarak<=1520))
            {merah=0;
                if (lurus==1)
                {
                    rata2();
                    label20->Text= String::Concat(lebar_s);
                }}}

        if(orang>50)kosongkan();
        if(SenPosY[10]<SenPosY[1]-20) kosongkan();
        if(size_avg==0) label9->Text =
String::Concat("Silahkan Tunggu...");
        if(size_avg!=0) label9->Text =
String::Concat("Selesai");
        if (kondisi ==rata)
        {
              average();
              kondisi=0;
              RandomIn[0]=lebar_s;
              RandomIn[1]=LSdepth;
              RandomIn[2]=RSdepth;
              RandomIn[3]=LSdepth - chestdepth;
              RandomIn[4]=LSdepth - bellydepth;
              NN_main();
              label21->Text =
String::Concat("",(size_avg));//12.ToString();
              kondisi=0;
              jalan=0;
              kategori();
             }
    }
    cvReleaseImage(&color);
    cvReleaseImage(&depthh);
    cvDestroyWindow("Color Image");
```

```
   cvDestroyWindow("Depth Image");
            }
private: System::Void timer1_Tick(System::Object^  sender,
System::EventArgs^  e) {
        }
private: System::Void button5_Click(System::Object^
sender, System::EventArgs^  e) {
            RandomIn[0]=lebar_s;
            RandomIn[1]=LSdepth;
            RandomIn[2]=RSdepth;
            RandomIn[3]=LSdepth - chestdepth;
            RandomIn[4]=LSdepth - bellydepth;
            NN_main();
MessageBox::Show(size_est.ToString());
        }

private: System::Void button3_Click_1(System::Object^
sender, System::EventArgs^  e) {
        }
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
            CompTime = GetTickCount();
        }
```

---Halaman ini sengaja dikosongkan---